

S2: Analysis of FAB eye tracking data with brms

I. S. Plank

2025-01-30

S2.1 Introduction

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

Some general settings

```
# number of simulations
nsim = 250

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

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] "tidyverse version 2.0.0"
## [1] "ggnetwork version 0.6.0"
## [1] "ggridges version 0.0.4"
## [1] "bayesplot version 1.11.1"
## [1] "SBC version 0.3.0.9000"
## [1] "rstatix version 0.7.2"
## [1] "logspline version 2.1.22"
## [1] "BayesFactor version 0.9.12.4.7"
## [1] "bayestestR version 0.15.0"
```

Preparation

First, we load the eye tracking data and combine it with demographic information including the diagnostic status of the subjects. Second, we preprocess the latencies of the saccades and divide them into saccades elicited by the cues and saccades elicited by the target. To do so, we use the knowledge that latencies below 100ms are extremely unlikely and local minima in the density function. We also load in the behavioural data again to be able to use reaction times for further correlational analyses.

```
# load the data
load("FAB_data.RData")

# combine both behavioural datasets
df.fab = rbind(df.fab, df.exp)
```

```

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

# aggregate the behavioural data disregarding stimuli and trials
df.fab.agg = rbind(df.fab, df.exp) %>%
  group_by(subID, diagnosis, cue) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  ) %>%
  group_by(subID, diagnosis) %>%
  # calculate subject-specific difference between cues
  summarise(
    rt.fab = combn(rt.cor, 2, FUN = function(x) x[1] - x[2])
  )

# remove participants without any data
df.sac = df.sac %>% filter(!is.na(lat))

# remove unbelievably short and extremely long saccades
df.sac = df.sac %>%
  filter(lat <= quantile(df.sac$lat, probs = 0.99) &
        lat > 100)

# divide into cue and target saccades
criticalpoints = function(density, threshold = 1){
  up = sapply(1:threshold, function(n) c(density$y[-seq(n)], rep(NA, n)))
  down = sapply(-1:-threshold,
                function(n) c(rep(NA, abs(n)),
                             density$y[-seq(length(density$y),
                                           length(density$y) - abs(n) + 1)]))
  a = cbind(density$y, up, down)
  minima = round(density$x[which(apply(a, 1, min) == a[,1])])
  maxima = round(density$x[which(apply(a, 1, max) == a[,1])])
  return(list(minima = minima, maxima = maxima))
}

points = criticalpoints(density(df.sac$lat))

# get the density of the latencies
dd = with(density(df.sac$lat), data.frame(x,y))

# find which point is the global minimum
lat.points = dd$y[points$minima]
idx = which.min(lat.points)

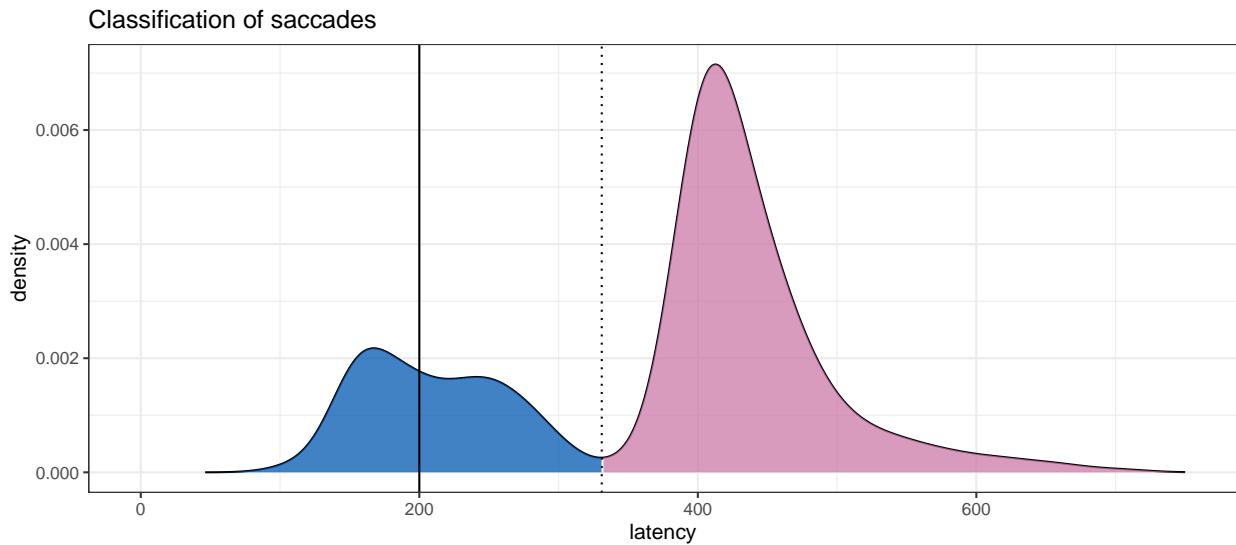
ggplot(dd, aes(x = x, y = y)) +
  geom_line() +
  geom_vline(xintercept = points$minima[idx], linetype=3) +
  geom_ribbon(data = subset(dd, x <= points$minima[idx]), aes(ymax = y, ymin = 0,
                                                             fill = custom.col[1], colour = NA, alpha = 0.75) +
  geom_ribbon(data = subset(dd, x >= points$minima[idx]), aes(ymax = y, ymin = 0,
                                                             fill = custom.col[2], colour = NA, alpha = 0.75) +
  geom_vline(xintercept = 200) +

```

```

  labs(title = "Classification of saccades", x = "latency", y = "density") +
  xlim(0, 750) +
  theme_bw()

```



The graph above shows the density of the latencies with zero on the x-axis being the onset of the cue. After 200ms, the cue disappears and the target is presented on the screen (solid line). We can see that there is a local minimum about 125ms after the target appears (dotted line). We can assume that saccades produced before this were in response to the cue (pink) and saccades after were in response to the target (green). Therefore, we divide the saccades accordingly. Then, we aggregate the data per subject and cue. Last, we set all predictors to sum contrasts.

```

# preprocess face saccade counts
df.cnt = df.sac %>%
  group_by(subID, diagnosis, dir_face) %>%
  summarise(
    n.face = n()
  )

# preprocess cue saccade counts
df.cnt.cue = df.sac %>%
  filter(lat <= points$minima[idx]) %>%
  group_by(subID, diagnosis, dir_face) %>%
  summarise(
    n.cue = n()
  )

# preprocess target saccade counts
df.cnt.tar = df.sac %>%
  filter(lat > points$minima[idx] & dir_target) %>%
  group_by(subID, diagnosis, cue) %>%
  summarise(
    n.tar = n()
  )

# add a zero if no saccades were produced

```

```

subID      = rep(as.character(unique(df.sac$subID)),
                 each = length(unique(df.sac$dir_face)))
dir_face  = rep(as.character(unique(df.sac$dir_face)),
                 times = length(unique(df.sac$subID)))
df.cnt = merge(df.cnt, data.frame(subID, dir_face), all = T) %>%
  mutate(
    n.face = if_else(is.na(n.face), 0, n.face)
  ) %>%
  # merge with behavioural data
  merge(., df.fab.agg) %>%
  mutate_if(is.character, as.factor)
df.cnt.cue = merge(df.cnt.cue, data.frame(subID, dir_face), all = T) %>%
  mutate(
    n.cue = if_else(is.na(n.cue), 0, n.cue)
  ) %>%
  # merge with behavioural data
  merge(., df.fab.agg) %>%
  mutate_if(is.character, as.factor)
cue      = rep(as.character(unique(df.sac$cue)), times = length(unique(df.sac$subID)))
df.cnt.tar = merge(df.cnt.tar, data.frame(subID, cue), all = T) %>%
  mutate(
    n.tar = if_else(is.na(n.tar), 0, n.tar)
  ) %>%
  # merge with behavioural data
  merge(., df.fab.agg) %>%
  mutate_if(is.character, as.factor)

# preprocess target latencies
df.lat = df.sac %>%
  # only keep latencies during cue
  filter(lat > points$minima[idx]) %>%
  # only keep the first latency of each trial
  group_by(subID, diagnosis, trl, cue) %>%
  # only keep the first of the remaining saccades
  filter(sac_trl == min(sac_trl)) %>%
  merge(., df.fab) %>%
  mutate_if(is.character, as.factor)

# add whether or not saccade to df.fab
df.fab = df.fab %>%
  merge(., df.lat %>% select(subID, trl, lat), all.x = T) %>%
  mutate(
    sac = if_else(is.na(lat), F, T)
  ) %>% select(-lat)

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

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

```

```

contrasts(df.lat$diagnosis) = contr.sum(4)
contrasts(df.lat$diagnosis)

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

contrasts(df.cnt.tar$cue) = contr.sum(2)
contrasts(df.cnt.tar$cue)

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

contrasts(df.cnt.tar$diagnosis) = contr.sum(4)
contrasts(df.cnt.tar$diagnosis)

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

contrasts(df.cnt.cue$dir_face) = contr.sum(2)
contrasts(df.cnt.cue$dir_face)

##      [,1]
## FALSE   1
## TRUE   -1

contrasts(df.cnt.cue$diagnosis) = contr.sum(4)
contrasts(df.cnt.cue$diagnosis)

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

contrasts(df.cnt$dir_face) = contr.sum(2)
contrasts(df.cnt$dir_face)

##      [,1]
## FALSE   1
## TRUE   -1

contrasts(df.cnt$diagnosis) = contr.sum(4)
contrasts(df.cnt$diagnosis)

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

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

```

S2.2 Number of saccades towards face during trial

First, we are going to assess the saccades that are produced in the direction of the face throughout the full trial: cue and target presentation. Based on Entzmann et al. (2021), we hypothesised that COMP participants produce more saccades towards the face than towards the object cues during the trials.

Specify the model

Since we are counting the number of saccades, we use a poisson distribution for our model. We add an group-level intercept for each subject, and assess the influence of the predictors diagnostic status and whether the saccade was directed towards the side of the face or object as well as the interaction. We set our priors very wide because we do not have strong prior assumptions apart from people producing fewer saccades than there are trials.

```

code = "CNT"

# set the formula
f.cnt = brms::bf(n.face ~ diagnosis * dir_face + (1 | subID))

# set priors based on study design
priors = c(
  prior(normal(3, 1.5), class = Intercept),
  prior(normal(0, 1.0), class = sd),
  prior(normal(0, 1.0), class = b)
)

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

# 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
  set.seed(2468)
  gen = SBC_generator_brms(f.cnt, data = df.cnt, prior = priors,
    thin = 50, warmup = 10000, refresh = 2000,
    generate_lp = TRUE, family = poisson(), init = 0.1)
}
```

```

bck = SBC_backend_brms_from_generator(gen, chains = 4, thin = 1,
  warmup = warm, iter = iter)
dat = generate_datasets(gen, nsim)
saveRDS(dat, file.path(cache_dir, sprintf("dat_%s.rds", code)))
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 5 of 250 simulations had at least one parameter that had an rhat of at least 1.05, and only 1 models had divergent samples. This suggests that this model performs well.

Prior predictive checks

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

```

# get the true values
truePars = dat[["variables"]]

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

# set very large data points to a value of 432
dvfakematH = dvfakemat;
dvfakematH[dvfakematH > 432] = 432
# compute one histogram per simulated data-set
breaks = seq(0, max(dvfakematH, na.rm=T), length.out = 100)
binwidth = round(breaks[2] - breaks[1])
breaks = seq(0, max(dvfakematH, na.rm=T), binwidth)
histmat = matrix(NA, ncol = nrow(truePars) + binwidth, nrow = length(breaks)-1)
for (i in 1:nrow(truePars)) {
  histmat[,i] = hist(dvfakematH[,i], breaks = breaks, plot = F)$counts
}
# for each bin, compute quantiles across histograms
probs = seq(0.1, 0.9, 0.1)
quantmat= as.data.frame(matrix(NA, nrow=dim(histmat)[1], ncol = length(probs)))
names(quantmat) = paste0("p", probs)
for (i in 1:dim(histmat)[1]) {
  quantmat[i,] = quantile(histmat[i,], p = probs, na.rm = T)
}
quantmat$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 = "Prior predictive distribution", y = "", x = "number of saccades") +
theme_bw()

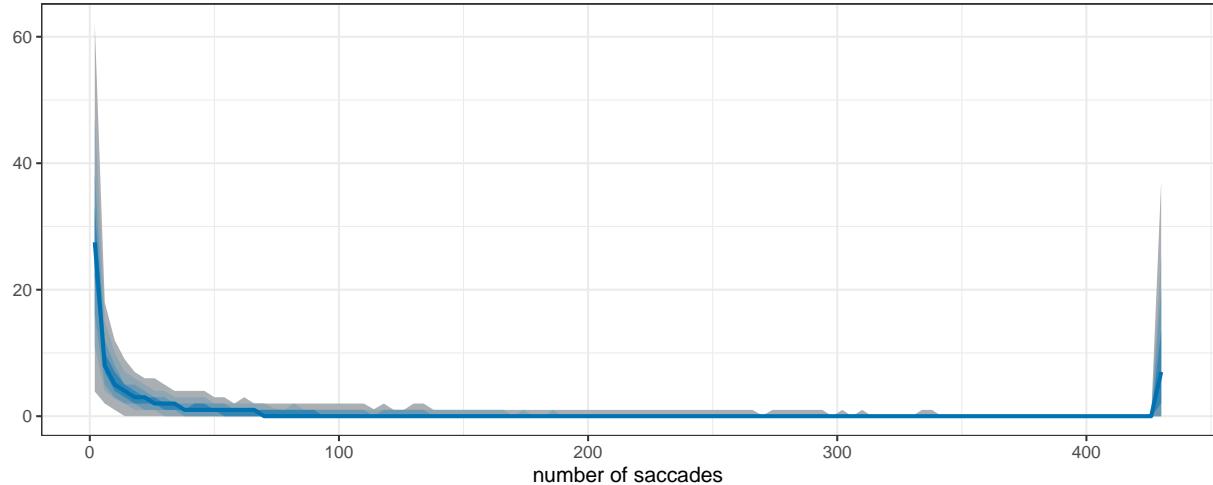
tmpM = apply(dvfakemathH, 2, mean) # mean
tmpSD = apply(dvfakemathH, 2, sd)
p2 = ggplot() +
  stat_bin(aes(x = tmpM), fill = c_dark) +
  labs(x = "Mean number of saccades", title = "Means of simulated data") +
  theme_bw()
p3 = ggplot() +
  stat_bin(aes(x = tmpSD), fill = c_dark) +
  labs(x = "SD number of saccades", title = "Standard deviations of simulated data") +
  theme_bw()

p = ggarrange(p1,
  ggarrange(p2, p3, ncol = 2, labels = c("B", "C")),
  nrow = 2, labels = "A")
annotate_figure(p, top = text_grob("Prior predictive checks", face = "bold", size = 14))

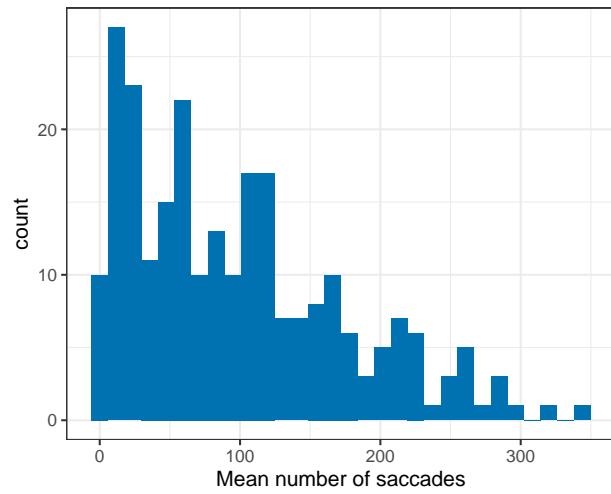
```

Prior predictive checks

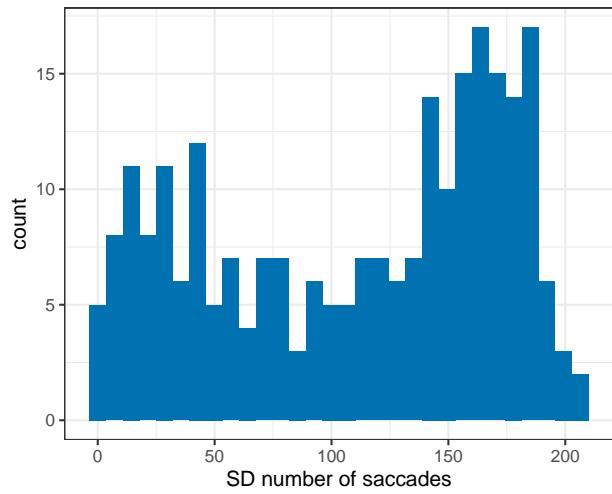
A Prior predictive distribution



B Means of simulated data



C Standard deviations of simulated data



Since our priors were set very wide, we do get wide prior predictive distributions. We accept this and continue with our checks.

Computational faithfulness and model sensitivity

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

# plot SBC with functions from the SBC package focusing on population-level parameters
```

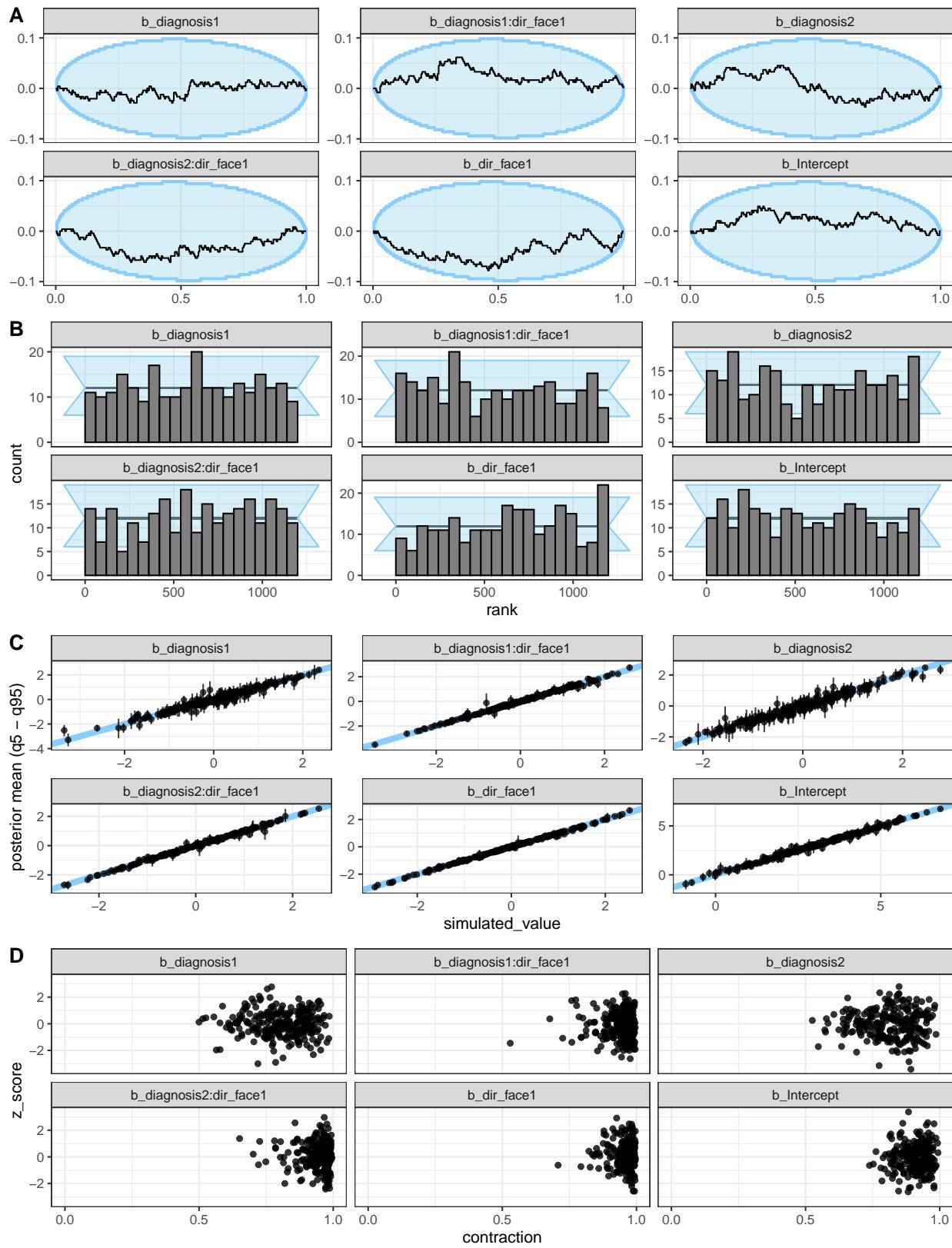
```

df.results.b = df.results %>%
  filter(substr(variable, 1, 2) == "b_") %>%
  filter(!(sim_id %in% check$sim_id)) %>%
  ungroup() %>%
  mutate(
    max_rank = max(rank)
  )
p1 = plot_ecdf_diff(df.results.b) + theme_bw() + theme(legend.position = "none") +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p2 = plot_rank_hist(df.results.b, bins = 20) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p3 = plot_sim_estimated(df.results.b, alpha = 0.75) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p4 = plot_contraction(df.results.b,
                      prior_sd = setNames(
                        c(as.numeric(
                          gsub(".*, (.+)\\" .*", "\\\\"1",
                               priors[priors$class == "Intercept",]$prior)),
                        rep(
                          as.numeric(
                            gsub(".*, (.+)\\" .*", "\\\\"1",
                               priors[priors$class == "b",]$prior)),
                          length(unique(df.results.b$variable))-1)),
                        unique(df.results.b$variable))) +
  theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))

p = ggarrange(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))

```

Computational faithfulness and model sensitivity



Second, 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.

Third, 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). Despite our wide priors, the contraction shows a bit of a distribution which increases our trust that the wide priors are appropriate.

Posterior predictive checks

As the next step, we fit the model and check whether the chains have converged, which they seem to have. We then perform posterior predictive checks on the model using the bayesplot package.

```
# fit the model
set.seed(2468)
m.cnt = brm(f.cnt,
             df.cnt, prior = priors,
             iter = iter, warmup = warm,
             backend = "cmdstanr", threads = threading(8),
             file = "m_cnt-face",
             family = "poisson",
             save_pars = save_pars(all = TRUE)
            )
rstan::check_hmc_diagnostics(m.cnt$fit)

##
## Divergences:
## 0 of 12000 iterations ended with a divergence.

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

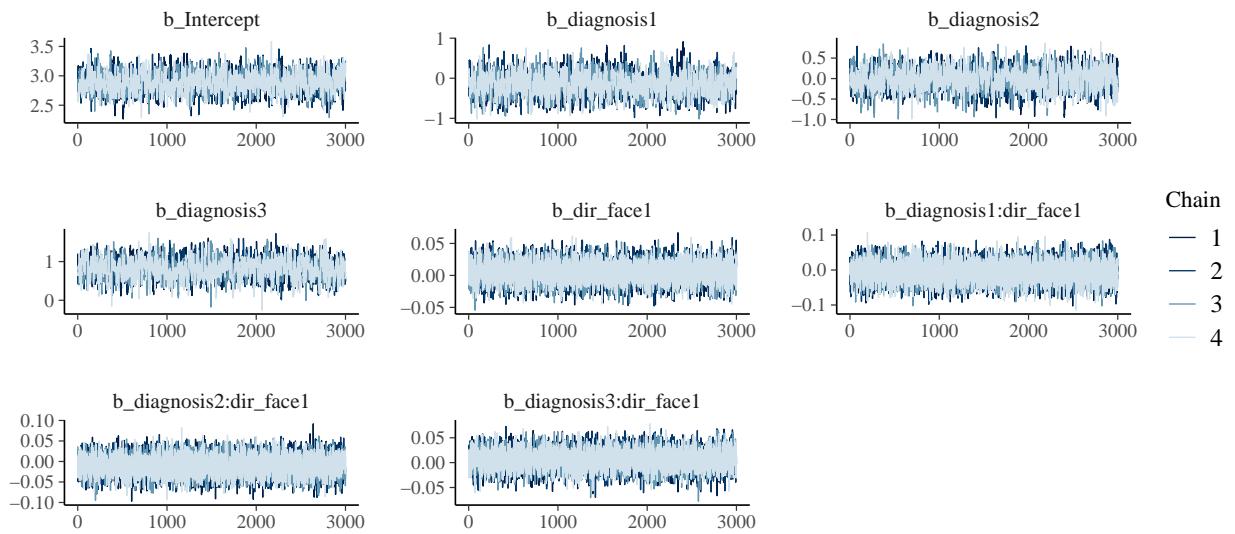
##
## Energy:
## E-BFMI indicated no pathological behavior.

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

## [1] 0

# check the trace plots
post.draws = as_draws_df(m.cnt)
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 divergent samples and no rhats that are higher or equal to 1.01. Therefore, we go ahead and perform our posterior predictive checks.

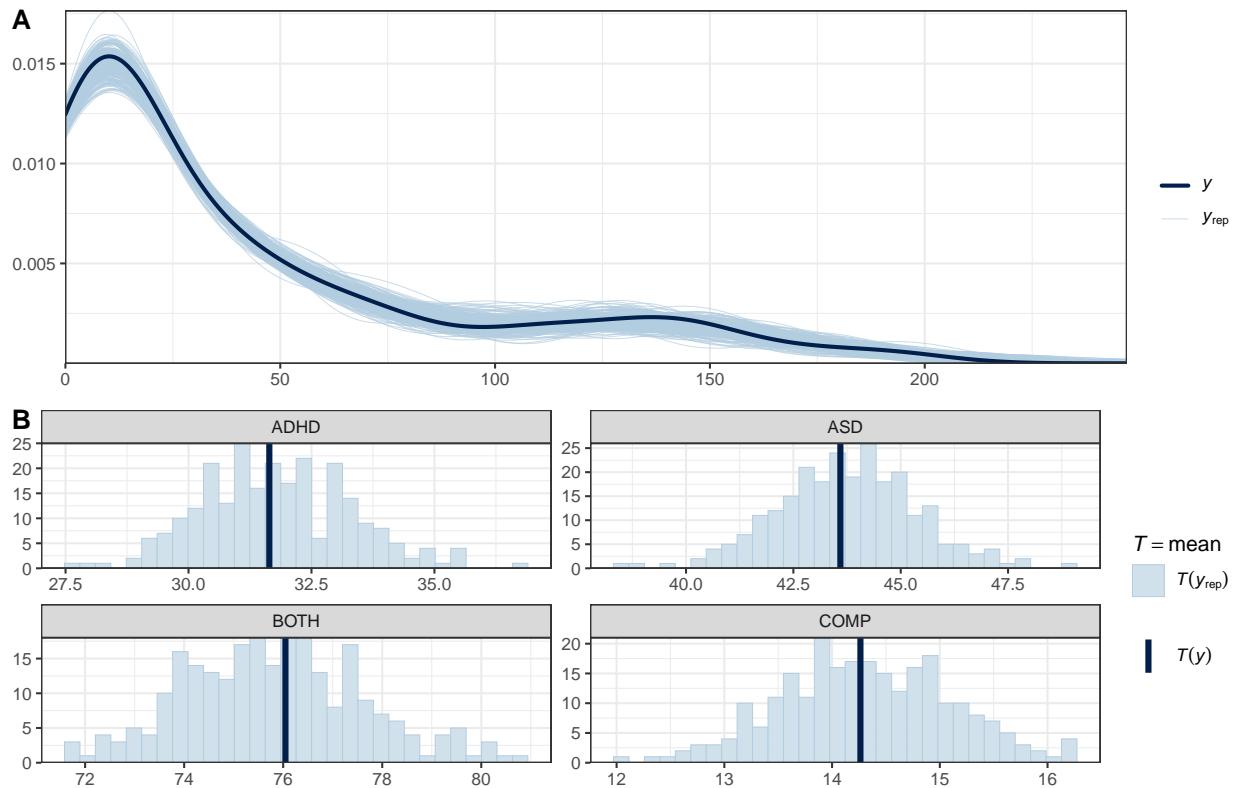
```
# get the posterior predictions
post.pred = posterior_predict(m.cnt, ndraws = nsim)

# check the fit of the predicted data compared to the real data
p1 = ppc_check(m.cnt, ndraws = nsim) +
  theme_bw()

# distributions of means and sds compared to the real values per group
p2 = ppc_stat_grouped(df.cnt$n.face, post.pred, df.cnt$diagnosis) +
  theme_bw()

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

Posterior predictive checks: number of saccades towards face



The predictions based on the model capture the data well. This further increases our trust in the model and we move on to interpret its parameter.

Inferences

Now that we are convinced that we can trust our model, we have a look at the model and its estimates.

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

```
## Family: poisson
## Links: mu = log
## Formula: n.face ~ diagnosis * dir_face + (1 | subID)
## Data: df.cnt (Number of observations: 147)
## Draws: 4 chains, each with iter = 4500; warmup = 1500; thin = 1;
##          total post-warmup draws = 12000
##
## Multilevel Hyperparameters:
## ~subID (Number of levels: 76)
##                               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)           1.32      0.12    1.11     1.58 1.00    1895     3363
## 
## Regression Coefficients:
##                               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                  2.90      0.16    2.59     3.21 1.00    1372
## diagnosis1                 -0.15      0.27   -0.67     0.38 1.00    1528
## diagnosis2                 -0.01      0.25   -0.51     0.48 1.01    1312
## diagnosis3                  0.82      0.24    0.34     1.29 1.00    1458
```

```

## dir_face1          0.01    0.02   -0.03    0.04 1.00    8146
## diagnosis1:dir_face1 -0.01    0.03   -0.06    0.05 1.00   10740
## diagnosis2:dir_face1 -0.01    0.02   -0.06    0.04 1.00    9289
## diagnosis3:dir_face1  0.01    0.02   -0.03    0.04 1.00    8029
##                                     Tail_ESS
## Intercept                  2549
## diagnosis1                 2591
## diagnosis2                 2437
## diagnosis3                 2562
## dir_face1                  9054
## diagnosis1:dir_face1      9223
## diagnosis2:dir_face1      8669
## diagnosis3:dir_face1      8769
##
## 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.cnt = as_draws_df(m.cnt) %>%
  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,
    b_dir_faceTRUE = - b_dir_face1,
    COMP       = b_Intercept + b_COMP
  )

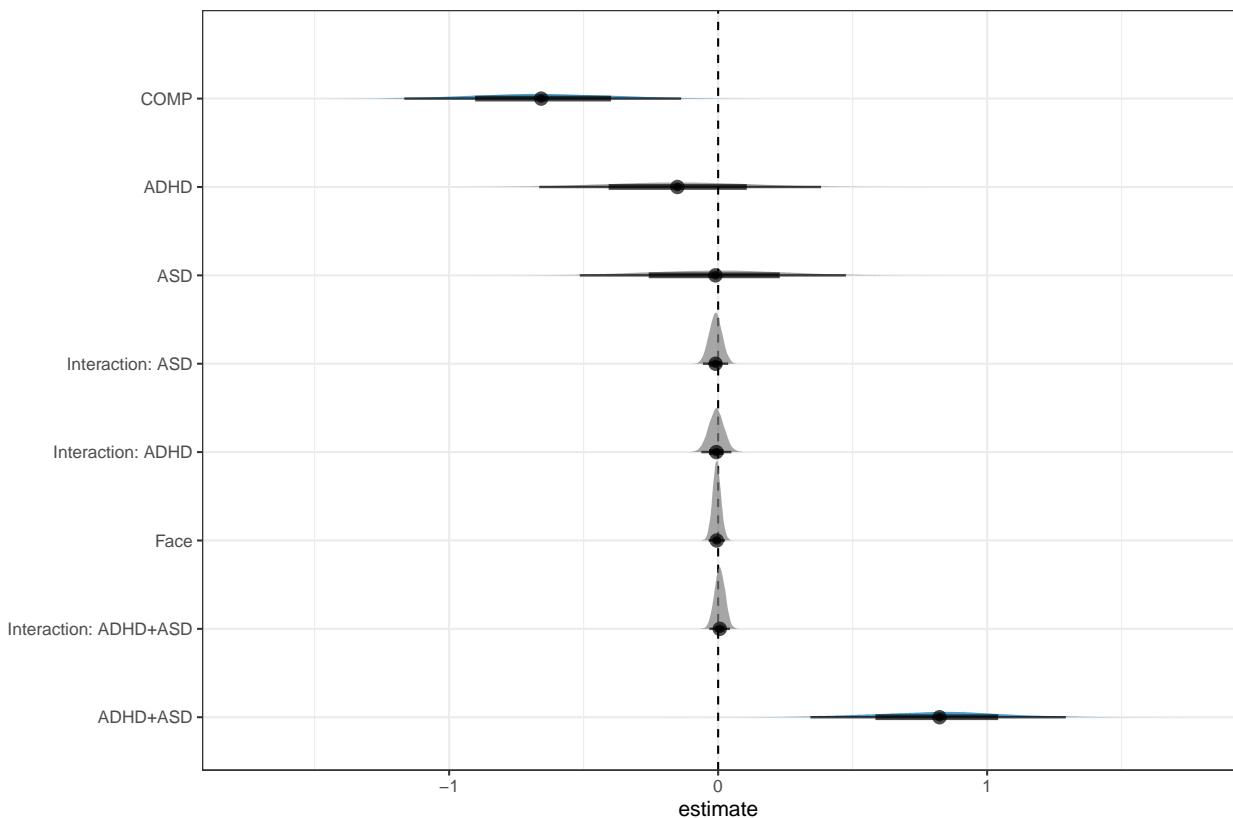
# plot the posterior distributions
df.m.cnt %>%
  select(starts_with("b_")) %>%
  pivot_longer(cols = starts_with("b_"), names_to = "coef", values_to = "estimate") %>%
  filter(coef != "b_Intercept" & coef != "b_dir_face1") %>%
  mutate(
    coef = case_match(coef,
      "b_diagnosis1" ~ "ADHD",
      "b_diagnosis2" ~ "ASD",
      "b_diagnosis3" ~ "ADHD+ASD",
      "b_COMP"       ~ "COMP",
      "b_dir_faceTRUE" ~ "Face",
      "b_diagnosis1:dir_face1" ~ "Interaction: ADHD",
      "b_diagnosis2:dir_face1" ~ "Interaction: ASD",
      "b_diagnosis3:dir_face1" ~ "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")

```



```

# H2a: COMP: face > object
h2a = hypothesis(m.cnt, "0 > dir_face1 - diagnosis1:dir_face1 - diagnosis2:dir_face1 - diagnosis3:dir_f
h2a

## Hypothesis Tests for class b:
##                               Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(dir_face1-di... > 0     -0.02      0.04    -0.09     0.06      0.57
##   Post.Prob Star
## 1      0.36
## ---
## 'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 95%;
## for two-sided hypotheses, the value tested against lies outside the 95%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.

# explore difference in saccades between COMP and BOTH
e1 = hypothesis(m.cnt, "0 < diagnosis1 + diagnosis2 + 2*diagnosis3",
                alpha = 0.025)
e1

```

```

## Hypothesis Tests for class b:
##                               Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(diagnosis1+d... < 0      -1.47      0.41     -2.28     -0.64      2999
##   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.
# explore difference in saccades between ADHD and BOTH
e2 = hypothesis(m.cnt, "0 > diagnosis1 - diagnosis3",
                 alpha = 0.025)
e2

## Hypothesis Tests for class b:
##                               Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(diagnosis1-d... > 0      0.97      0.41      0.15     1.78      91.31
##   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.
# explore difference in saccades between ASD and BOTH
e3 = hypothesis(m.cnt, "0 > diagnosis2 - diagnosis3",
                 alpha = 0.025)
e3

## Hypothesis Tests for class b:
##                               Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(diagnosis2-d... > 0      0.83      0.4       0.06     1.61      59.91
##   Post.Prob Star
## 1      0.98      *
## ---
## '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.
# check equivalence of the groups that seem similar
equ.cnt =
  equivalence_test(df.m.cnt %>%
    mutate(`b_ADHD-ASD` = b_diagnosis1 - b_diagnosis2,
          `b_COMP-ASD` = -b_diagnosis1 - 2*b_diagnosis2 - b_diagnosis3,
          `b_COMP-ADHD` = -2*b_diagnosis1 - b_diagnosis2 - b_diagnosis3
    ) %>%
    select(contains('~-')))
equ.cnt

## # Test for Practical Equivalence
##
## ROPE: [-0.10 0.10]
##

```

```

## Parameter |      H0 | inside ROPE |      95% HDI
## -----
## ADHD-ASD | Undecided |    18.34 % | [-0.97, 0.76]
## COMP-ASD  | Undecided |     6.56 % | [-1.45, 0.20]
## COMP-ADHD | Undecided |    10.30 % | [-1.35, 0.34]

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

# calculate our difference columns
df.est = df.est %>%
  mutate(
    h2a = COMP_TRUE - COMP_FALSE
  )

```

Our hypothesis was not confirmed: there was no difference between the number of saccades in the direction of the face compared to the direction of the object in our COMP group (CI of COMP(face) - COMP(object): -2.19 to 1.48, posterior probability = 36.16%).

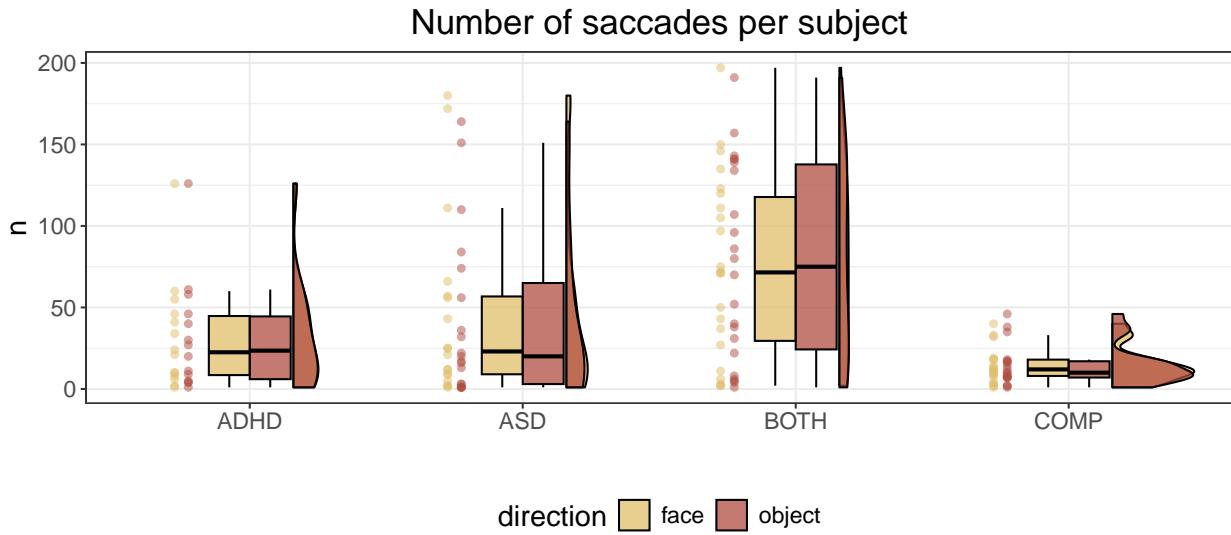
Plots

As a next step, we can now finally plot our data.

```

# rain cloud plot
df.cnt %>%
  mutate(direction = if_else(dir_face == T, "face", "object")) %>%
  ggplot(aes(diagnosis, n.face, fill = direction, colour = direction)) +
  geom_rain(rain.side = 'r',
  boxplot.args = list(color = "black", outlier.shape = NA, show_guide = FALSE, alpha = 0.75),
  violin.args = list(color = "black", outlier.shape = NA, alpha = 0.75),
  boxplot.args.pos = list(
    position = ggpp::position_dodgeNudge(x = 0, width = 0.3), width = 0.3
  ),
  point.args = list(show_guide = FALSE, alpha = .5),
  violin.args.pos = list(
    width = 0.6, position = position_nudge(x = 0.16)),
  point.args.pos = list(position = ggpp::position_dodgeNudge(x = -0.25, width = 0.1))) +
  scale_fill_manual(values = custom.col2) +
  scale_color_manual(values = custom.col2) +
  labs(title = "Number of saccades per subject", x = "", y = "n") +
  theme_bw() +
  theme(legend.position = "bottom",
        plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal",
        text = element_text(size = 15))

```



Bayes factor analysis

To complement our hypothesis testing using `brms::hypothesis()`, we perform a Bayes Factor analysis with models excluding some of our population-level predictors.

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

# 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(6633)
  m.cnt.bf = brm(f.cnt,
    df.cnt, prior = priors,
    iter = 40000, warmup = 10000,
    backend = "cmdstanr", threads = threading(8),
    file = "m_cnt_bf",
    family = "poisson",
    save_pars = save_pars(all = TRUE)
  )
}
```

```

}

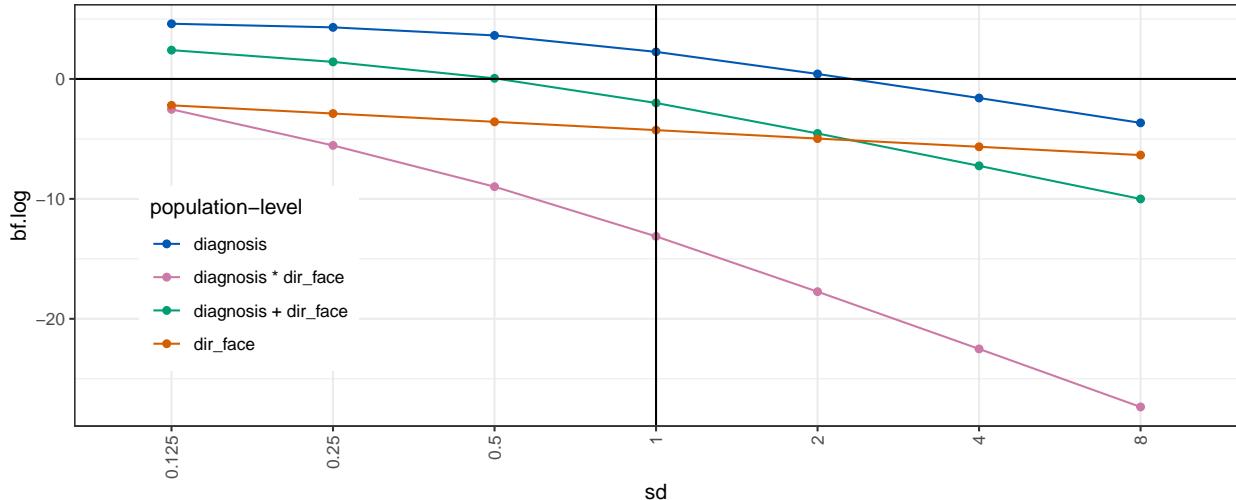
# loop through them
for (pr.desc in pr.descriptions) {
  tryCatch({
    # use the function
    bf_sens_2int(m.cnt.bf, "diagnosis", "dir_face", pr.desc,
      main.code, # prefix for all models and MLL
      file.path(log_dir, "log_FAB_bf.txt"), # log file
      sense_dir, # where to save the models and MLL
      reps = 5
    )
  },
  error = function(err) {
    message(sprintf("Error for %s: %s", pr.desc, err))
  }
)
}

# read in the results
df.cnt.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.cnt.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") +
  #facet_wrap(. ~ `population-level`, scales = "free_y") +
  scale_colour_manual(values = custom.col) +
  theme_bw() +
  theme(legend.position = c(0.15,0.35),
    axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
}

```

Sensitivity analysis with the intercept–only model as reference



```
# print the BFs based on chosen priors
kable(df.cnt.bf %>% filter(priors == "chosen") %>% select(-priors) %>%
  filter(`population-level` != "1") %>% arrange(desc(bf.log)), digits = 3)
```

population-level	bf.log
diagnosis	2.259
diagnosis + dir_face	-2.002
dir_face	-4.266
diagnosis * dir_face	-13.125

The Bayes Factor analysis revealed the model only including the main effect diagnostic status outperforming the intercept-only model. With wider priors, the Bayes Factor of this model also indicates worse fit for our data than the intercept-only model. For our chosen priors though, there is moderate evidence in favour of this model compared to the intercept-only model (diagnostic groups: $\log(BF) = 2.259$). The intercept-only model consistently outperforms the other models (saccade direction: $\log(BF) = -4.266$; diagnostic group and saccade direction: $\log(BF) = -2.002$; full model: $\log(BF) = -13.125$). These results suggest that diagnostic status influences the amount of saccades made, however, this effect is independent of the direction.

S2.3 Saccade latencies

Next, we focus on the latencies of the saccades that are produced during the presentation of the targets to assess whether cue type, diagnostic group or their interaction influence latencies. Based on Guillon et al. (2014), we hypothesised that ASD participants show an increased latency compared to COMP participants when producing saccades towards targets appearing at the location of a face.

Full model

We start with the model containing all latencies of saccades produced during the target presentation. We choose a shifted lognormal distribution because saccade latencies below 100ms are very unlikely. Additionally, latencies are determined based on the onset of the cue presentation (200ms).

Setting up and assessing the model

```

code = "LAT"

# set the formula
f.lat = brms::bf(lat ~ diagnosis * cue + (cue | subID) + (diagnosis * cue | stm))

# set weakly informative priors
priors = c(
  prior(normal(5, 0.75), class = Intercept),
  prior(normal(0, 0.25), class = sd),
  prior(normal(0, 0.25), class = b),
  prior(normal(0.5, 0.50), class = sigma),
  prior(normal(350, 50.00), class = ndt), # threshold between target and cue saccades
  prior(lkj(2), class = cor)
)

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

if (file.exists(file.path(cache_dir, paste0("df_res_", code, ".rds")))) {
  # load in the results of the SBC
  df.results = readRDS(file.path(cache_dir, paste0("df_res_", code, ".rds")))
  df.backend = readRDS(file = file.path(cache_dir, paste0("df_div_", code, ".rds")))
  dat = readRDS(file = file.path(cache_dir, paste0("dat_", code, ".rds")))
} else {
  # create the data and the results
  set.seed(2468)
  gen = SBC_generator_brms(f.lat, data = df.lat, prior = priors,
                           family = "shifted_lognormal",
                           thin = 50, warmup = 10000, refresh = 2000,
                           generate_lp = TRUE)
  dat = generate_datasets(gen, nsim)
  saveRDS(dat, file = file.path(cache_dir, paste0("dat_", code, ".rds")))
  backend = SBC_backend_brms_from_generator(gen, chains = 4, thin = 1,
                                             warmup = 1000, iter = 3000)
  results = compute_SBC(dat, backend,
                        cache_mode = "results",
                        cache_location = file.path(cache_dir, paste0("res_", code)))
  saveRDS(results$stats,
          file = file.path(cache_dir, paste0("df_res_", code, ".rds")))
  saveRDS(results$backend_diagnostics,
          file = file.path(cache_dir, paste0("df_div_", code, ".rds")))
}

```

We start by investigating the rhats and the number of divergent samples. This shows that 0 of 250 simulations had at least one parameter that had an rhat of at least 1.05, but 1 models had divergent samples (mean number of samples of the simulations with divergent samples: 1). This suggests that this model performs well.

Prior predictive checks

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

```

# get the true values
truePars = dat[["variables"]]

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

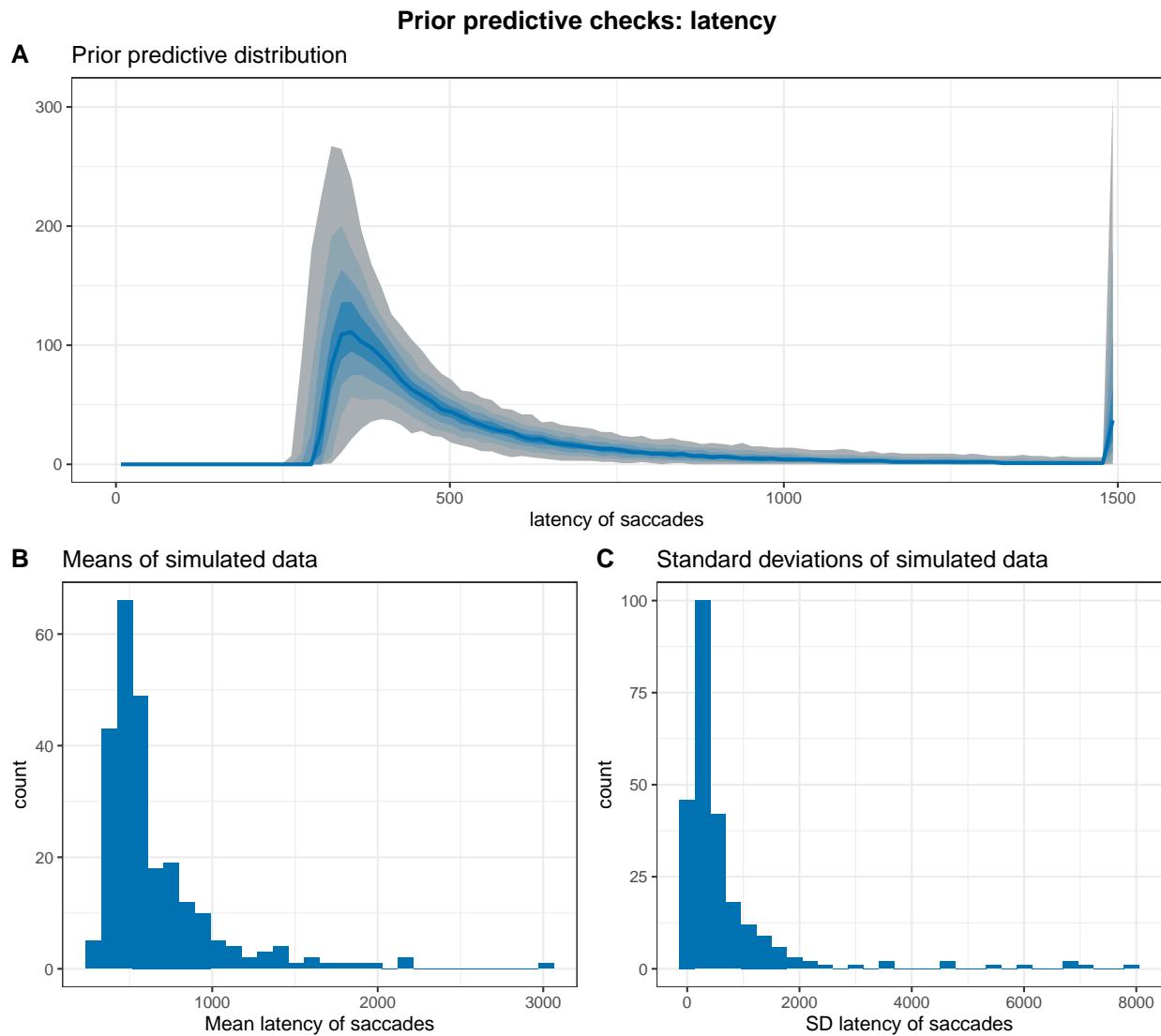
# set very large data points to a value of 1500
dvfakemath = dvfakemat;
dvfakemath[dvfakemath < 0] = 0
dvfakemath[dvfakemath > 1500] = 1500
# compute one histogram per simulated data-set
breaks = seq(0, 1500, length.out = 101)
binwidth = breaks[2] - breaks[1]
histmat = matrix(NA, ncol = nrow(truePars) + binwidth, nrow = length(breaks)-1)
for (i in 1:nrow(truePars)) {
  histmat[,i] = hist(dvfakemath[,i], breaks = breaks, plot = F)$counts
}
# for each bin, compute quantiles across histograms
probs = seq(0.1, 0.9, 0.1)
quantmat = as.data.frame(matrix(NA, nrow=dim(histmat)[1], ncol = length(probs)))
names(quantmat) = paste0("p", probs)
for (i in 1:dim(histmat)[1]) {
  quantmat[i,] = quantile(histmat[i,], p = probs, na.rm = T)
}
quantmat$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 = "Prior predictive distribution", y = "", x = "latency of saccades") +
  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 latency of saccades", title = "Means of simulated data") +
  theme_bw()
p3 = ggplot() +
  stat_bin(aes(x = tmpSD), fill = c_dark) +
  labs(x = "SD latency of saccades", title = "Standard deviations of simulated data") +
  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: latency",
face = "bold", size = 14))
```



First, we assess whether the simulated values fit our expectations of the distribution of the data. Previous literature has found that saccade latencies are around 200ms with few saccades being produced faster than 100ms. If we add 200ms from the cue presentation, this means we expect most latencies to be above 300ms and centered around 400ms. Our simulated datasets seem to capture this well.

Computational faithfulness and model sensitivity

```
# get simulation numbers with issues
des_rank = max(df.results$max_rank)
check = merge(df.results %>%
  group_by(sim_id) %>%
  summarise(
    rhat = max(rhat, na.rm = T),
    mean_rank = max(max_rank))
```

```

    ) %>%
  filter(rhat >= 1.05 | mean_rank < des_rank),
df.backend %>% filter(n_divergent > 0), all = T)

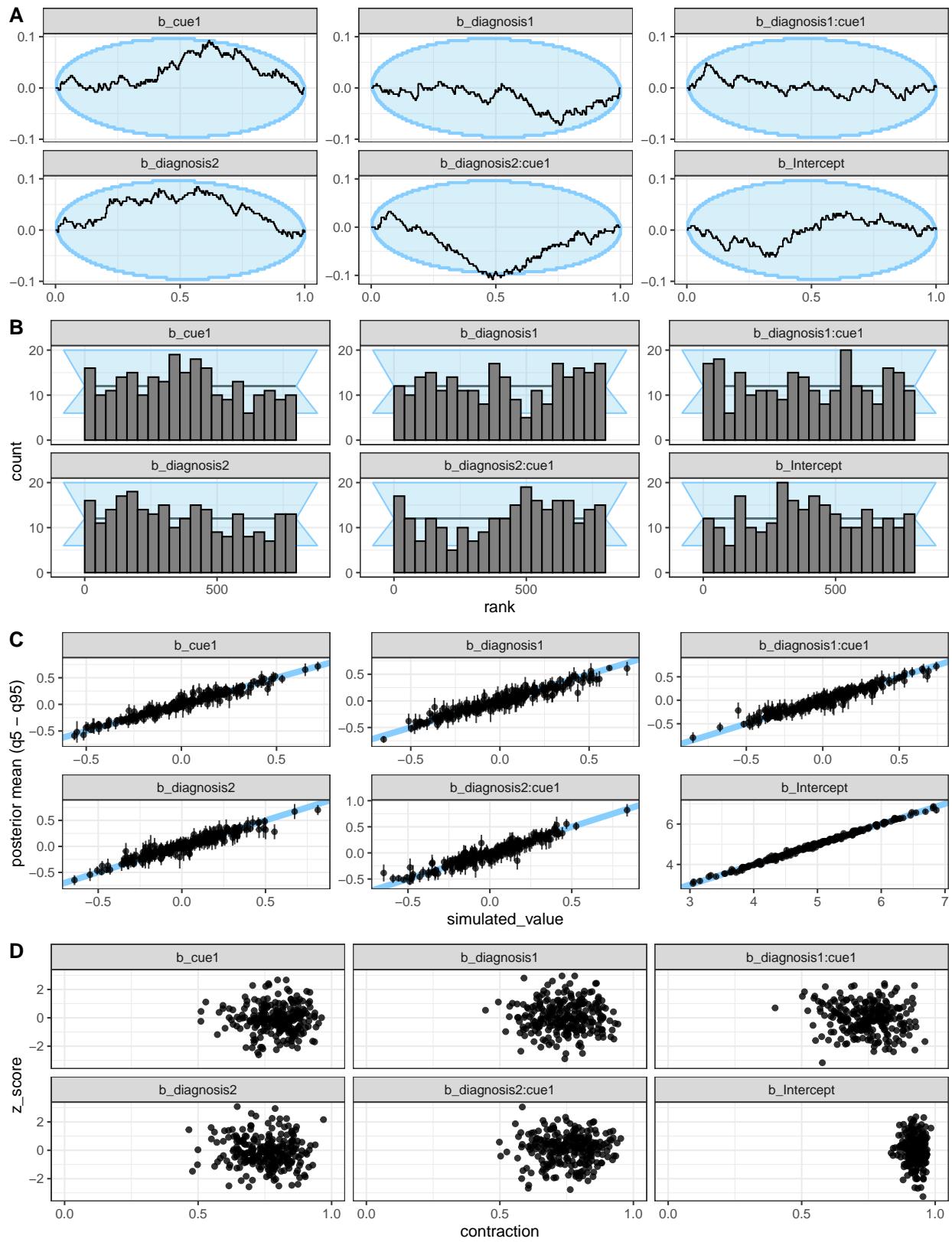
# plot SBC with functions from the SBC package focusing on population-level parameters

df.results.b = df.results %>%
  filter(substr(variable, 1, 2) == "b_") %>%
  filter(!sim_id %in% check$sim_id)) %>%
  ungroup() %>%
  mutate(
    max_rank = max(rank)
  )
p1 = plot_ecdf_diff(df.results.b) + theme_bw() + theme(legend.position = "none") +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p2 = plot_rank_hist(df.results.b, bins = 20) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p3 = plot_sim_estimated(df.results.b, alpha = 0.75) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p4 = plot_contraction(df.results.b,
  prior_sd = setNames(
    c(as.numeric(
      gsub(".*, (.+)\\".*", "\\\\"1",
      priors[priors$class == "Intercept",]$prior)),
    rep(
      as.numeric(
        gsub(".*, (.+)\\".*", "\\\\"1",
        priors[priors$class == "b",]$prior)),
      length(unique(df.results.b$variable))-1)),
    unique(df.results.b$variable))) +
  theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))

p = ggarrange(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))

```

Computational faithfulness and model sensitivity



Second, 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.

Third, 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). Both look acceptable.

Posterior predictive checks

As the next step, we fit the model and check whether the chains have converged, which they seem to have. We then perform posterior predictive checks on the model using the bayesplot package.

```
# fit the maximal model
set.seed(2587)
m.lat = brm(f.lat,
             df.lat, prior = priors,
             iter = iter, warmup = warm,
             backend = "cmdstanr", threads = threading(8),
             family = "shifted_lognormal",
             file = "m_lat",
             save_pars = save_pars(all = TRUE)
            )
rstan::check_hmc_diagnostics(m.lat$fit)

##
## Divergences:
## 0 of 8000 iterations ended with a divergence.

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

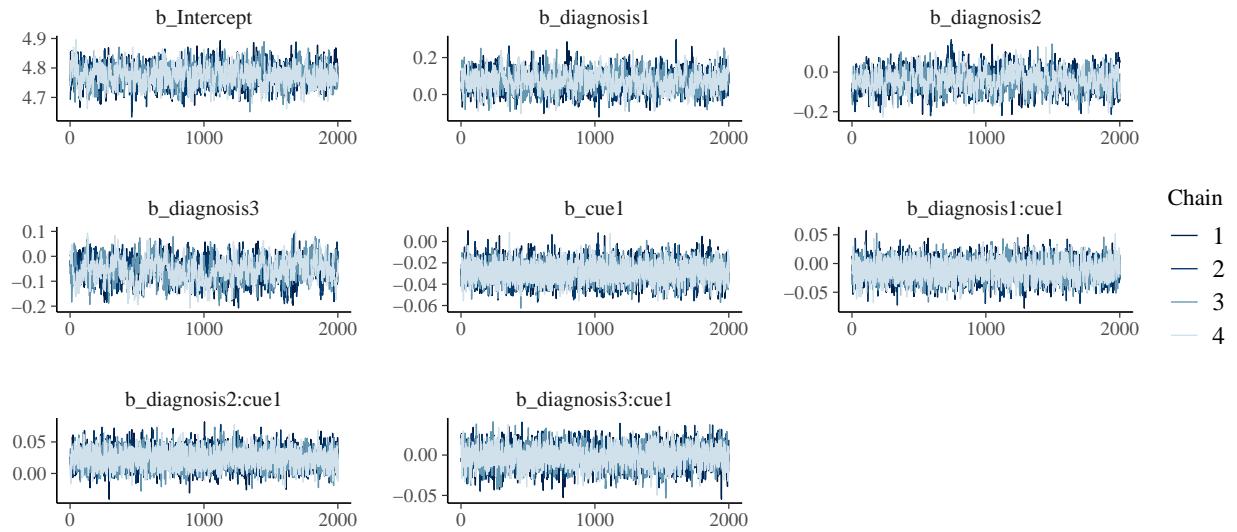
##
## Energy:
## E-BFMI indicated no pathological behavior.

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

## [1] 0

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



The model does not have any divergent transitions nor high rhats. The trace plots also look good, therefore, we move on to the posterior predictive checks.

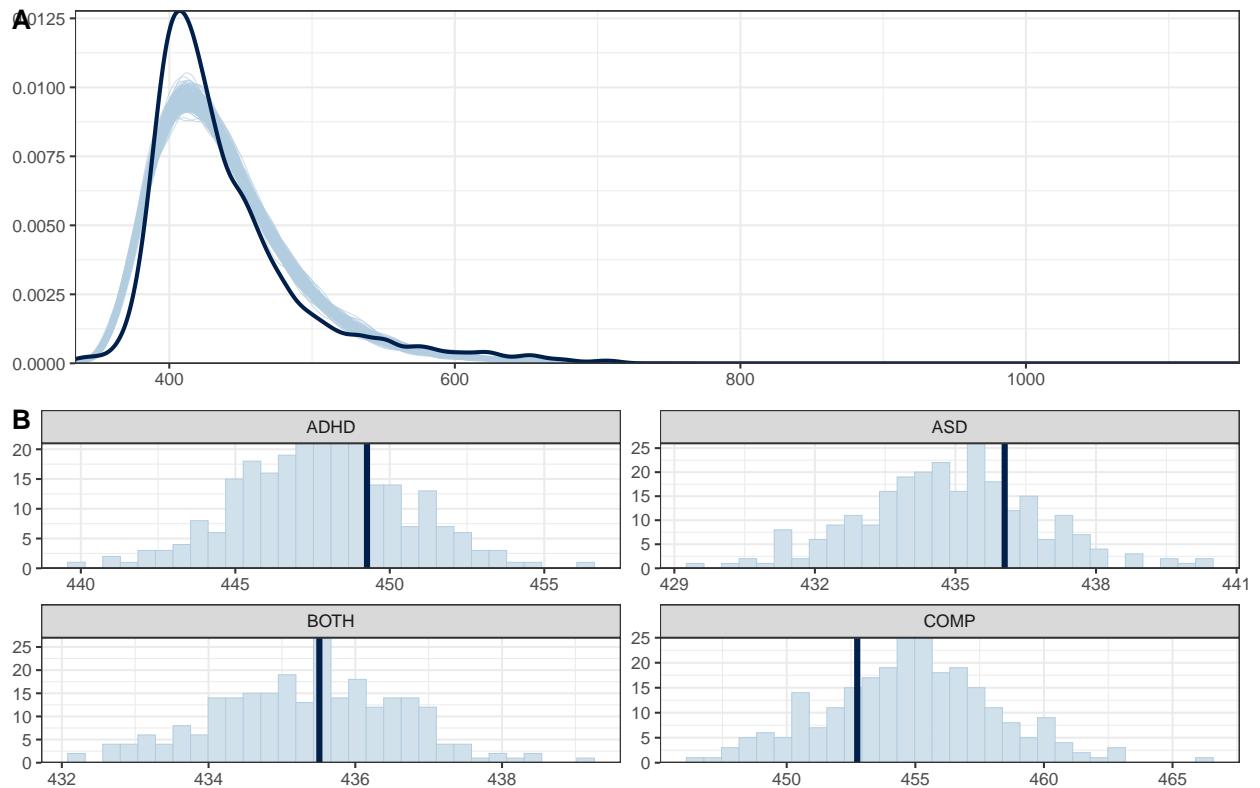
```
# get the posterior predictions
post.pred = posterior_predict(m.lat, ndraws = nsim)

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

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

Posterior predictive checks: latency



The simulated data based on the model does not fit our data very well: it is wider and seems to underestimate latencies for ADHD and COMP while overestimating for ASD with the dark blue line showing the mean of the actual dataset and the light blue bars showing the distribution of the predicted data.

Aggregated model

Since we want to base our inferences on the estimates, we go back to the drawing board and aggregate our data to see whether this resolves these issues.

Setting up and assessing the model

```
code = "LAT_agg"

# aggregate the data
df.lat.agg = df.lat %>%
  group_by(subID, cue, diagnosis) %>%
  summarise(lat = median(lat, na.rm = T))

# set the formula
f.lat = brms::bf(lat ~ diagnosis * cue + (1 | subID) )

# set weakly informative priors
priors = priors %>% filter(class != "cor")

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

```

if (file.exists(file.path(cache_dir, paste0("df_res_", code, ".rds")))) {
  # load in the results of the SBC
  df.results = readRDS(file.path(cache_dir, paste0("df_res_", code, ".rds")))
  df.backend = readRDS(file = file.path(cache_dir, paste0("df_div_", code, ".rds")))
  dat       = readRDS(file = file.path(cache_dir, paste0("dat_", code, ".rds")))
} else {
  # create the data and the results
  set.seed(2468)
  gen = SBC_generator_brms(f.lat, data = df.lat.agg, prior = priors,
                           family = "shifted_lognormal",
                           thin = 50, warmup = 10000, refresh = 2000,
                           generate_lp = TRUE)
  dat = generate_datasets(gen, nsim)
  saveRDS(dat, file = file.path(cache_dir, paste0("dat_", code, ".rds")))
  backend = SBC_backend_brms_from_generator(gen, chains = 4, thin = 1,
                                             warmup = 1000, iter = 3000)
  results = compute_SBC(dat, backend,
                        cache_mode      = "results",
                        cache_location = file.path(cache_dir, paste0("res_", code)))
  saveRDS(results$stats,
          file = file.path(cache_dir, paste0("df_res_", code, ".rds")))
  saveRDS(results$backend_diagnostics,
          file = file.path(cache_dir, paste0("df_div_", code, ".rds")))
}

```

We start by investigating the rhats and the number of divergent samples. This shows that 1 of 250 simulations had at least one parameter that had an rhat of at least 1.05, but 105 models had divergent samples (mean number of samples of the simulations with divergent samples: 7.21). This is something to look out for in the final model.

Prior predictive checks

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

```

# get the true values
truePars = dat[["variables"]]

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

# set very large data points to a value of 1500
dvfakematH = dvfakemat;
dvfakematH[dvfakematH < 0] = 0
dvfakematH[dvfakematH > 1500] = 1500
# compute one histogram per simulated data-set
breaks = seq(0, 1500, length.out = 101)
binwidth = breaks[2] - breaks[1]
histmat = matrix(NA, ncol = nrow(truePars) + binwidth, nrow = length(breaks)-1)
for (i in 1:nrow(truePars)) {
  histmat[,i] = hist(dvfakematH[,i], breaks = breaks, plot = F)$counts
}

```

```

# for each bin, compute quantiles across histograms
probs = seq(0.1, 0.9, 0.1)
quantmat= as.data.frame(matrix(NA, nrow=dim(histmat)[1], ncol = length(probs)))
names(quantmat) = paste0("p", probs)
for (i in 1:dim(histmat)[1]) {
  quantmat[i,] = quantile(histmat[i,], p = probs, na.rm = T)
}
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 = "Prior predictive distribution", y = "", x = "latency of saccades") +
  theme_bw()

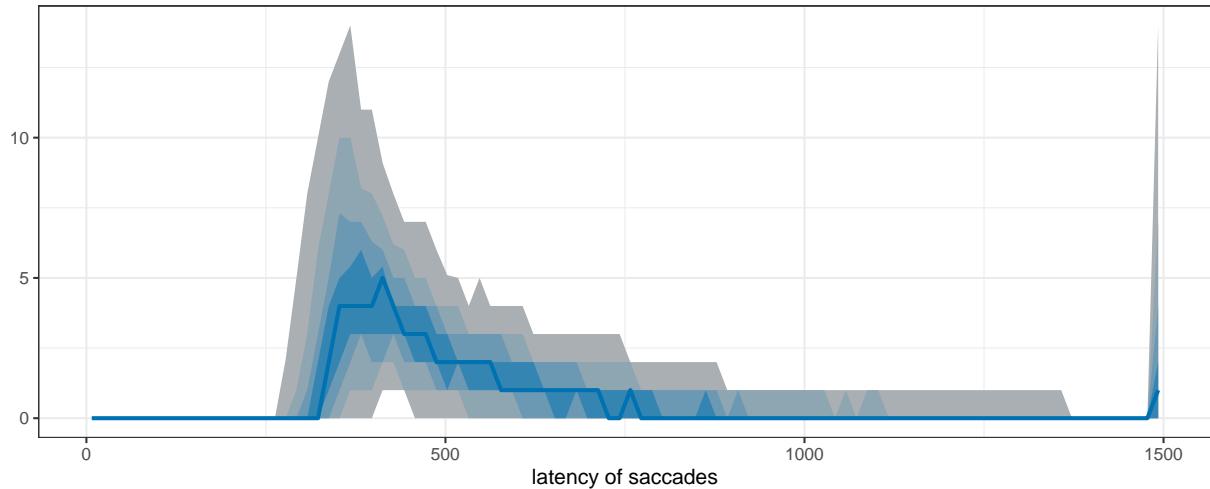
tmpM = apply(dvfakemat, 2, mean) # mean
tmpSD = apply(dvfakemat, 2, sd)
p2 = ggplot() +
  stat_bin(aes(x = tmpM), fill = c_dark) +
  labs(x = "Mean latency of saccades", title = "Means of simulated data") +
  theme_bw()
p3 = ggplot() +
  stat_bin(aes(x = tmpSD), fill = c_dark) +
  labs(x = "SD latency of saccades", title = "Standard deviations of simulated data") +
  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: latency",
  face = "bold", size = 14))

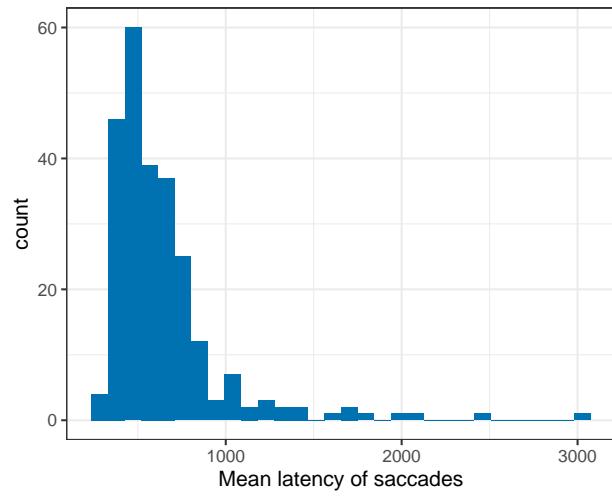
```

Prior predictive checks: latency

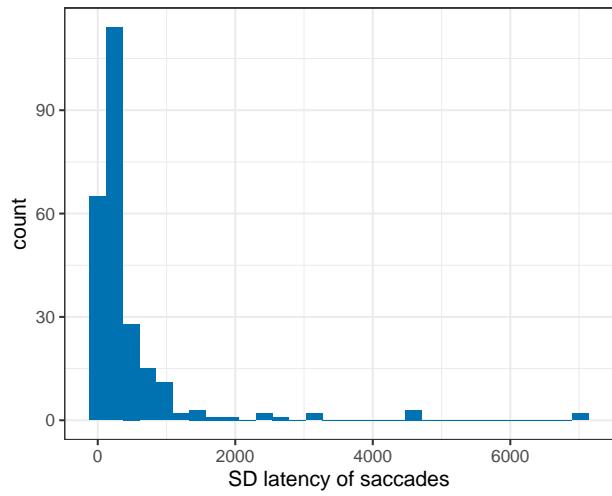
A Prior predictive distribution



B Means of simulated data



C Standard deviations of simulated data



Again, our simulated datasets seem to capture well what we know about saccade latencies.

Computational faithfulness and model sensitivity

```
# get simulation numbers with issues
des_rank = max(df.results$max_rank)
check = merge(df.results %>%
  group_by(sim_id) %>%
  summarise(
    rhat = max(rhat, na.rm = T),
    mean_rank = max(max_rank)
  ) %>%
  filter(rhat >= 1.05 | mean_rank < des_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 %>%
```

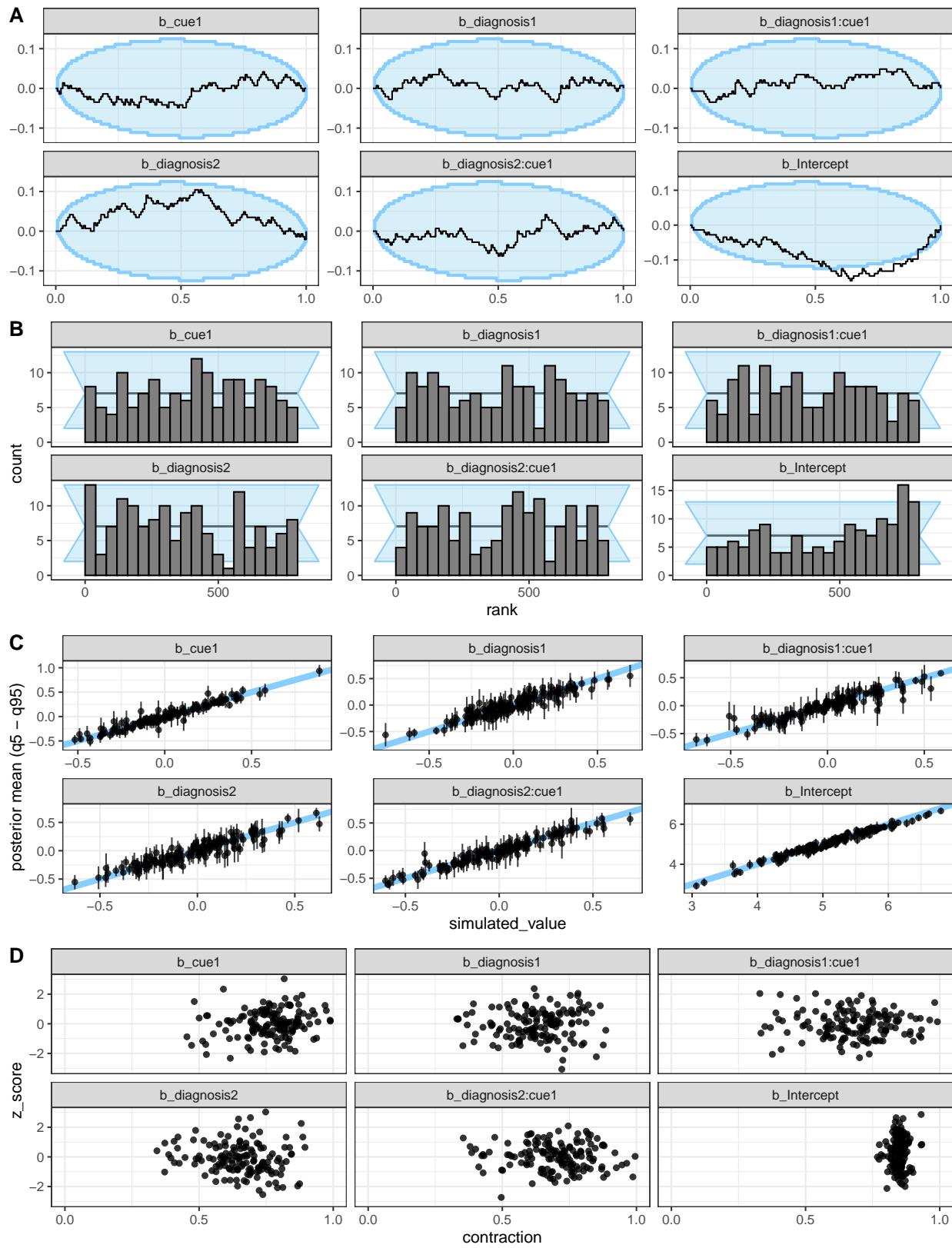
```

filter(substr(variable, 1, 2) == "b_") %>%
filter(!(sim_id %in% check$sim_id)) %>%
ungroup() %>%
mutate(
  max_rank = max(rank)
)
p1 = plot_ecdf_diff(df.results.b) + theme_bw() + theme(legend.position = "none") +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p2 = plot_rank_hist(df.results.b, bins = 20) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p3 = plot_sim_estimated(df.results.b, alpha = 0.75) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p4 = plot_contraction(df.results.b,
  prior_sd = setNames(
    c(as.numeric(
      gsub(".*, (.+)\\".*", "\\\\"1",
        priors[priors$class == "Intercept",]$prior)),
    rep(
      as.numeric(
        gsub(".*, (.+)\\".*", "\\\\"1",
          priors[priors$class == "b",]$prior)),
      length(unique(df.results.b$variable))-1)),
    unique(df.results.b$variable))) +
  theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))

p = ggarrange(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))

```

Computational faithfulness and model sensitivity



Second, 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.

Third, 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). Both look acceptable.

Posterior predictive checks

As the next step, we fit the model and check whether the chains have converged, which they seem to have. We then perform posterior predictive checks on the model using the bayesplot package.

```
# fit the maximal model
set.seed(7799)
m.lat = brm(f.lat,
             df.lat.agg, prior = priors,
             iter = iter, warmup = warm,
             backend = "cmdstanr", threads = threading(8),
             family = "shifted_lognormal",
             file = "m_lat_agg",
             save_pars = save_pars(all = TRUE)
            )
rstan::check_hmc_diagnostics(m.lat$fit)

##
## Divergences:
## 0 of 8000 iterations ended with a divergence.

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

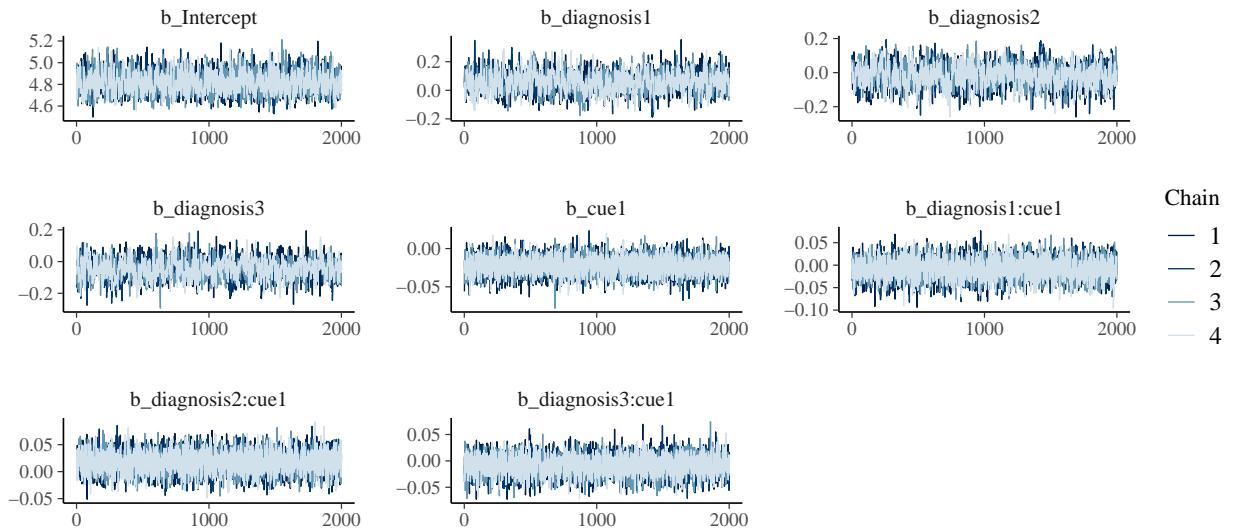
##
## Energy:
## E-BFMI indicated no pathological behavior.

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

## [1] 0

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



In the final model, there are very few divergent transitions, so we continue to assess it.

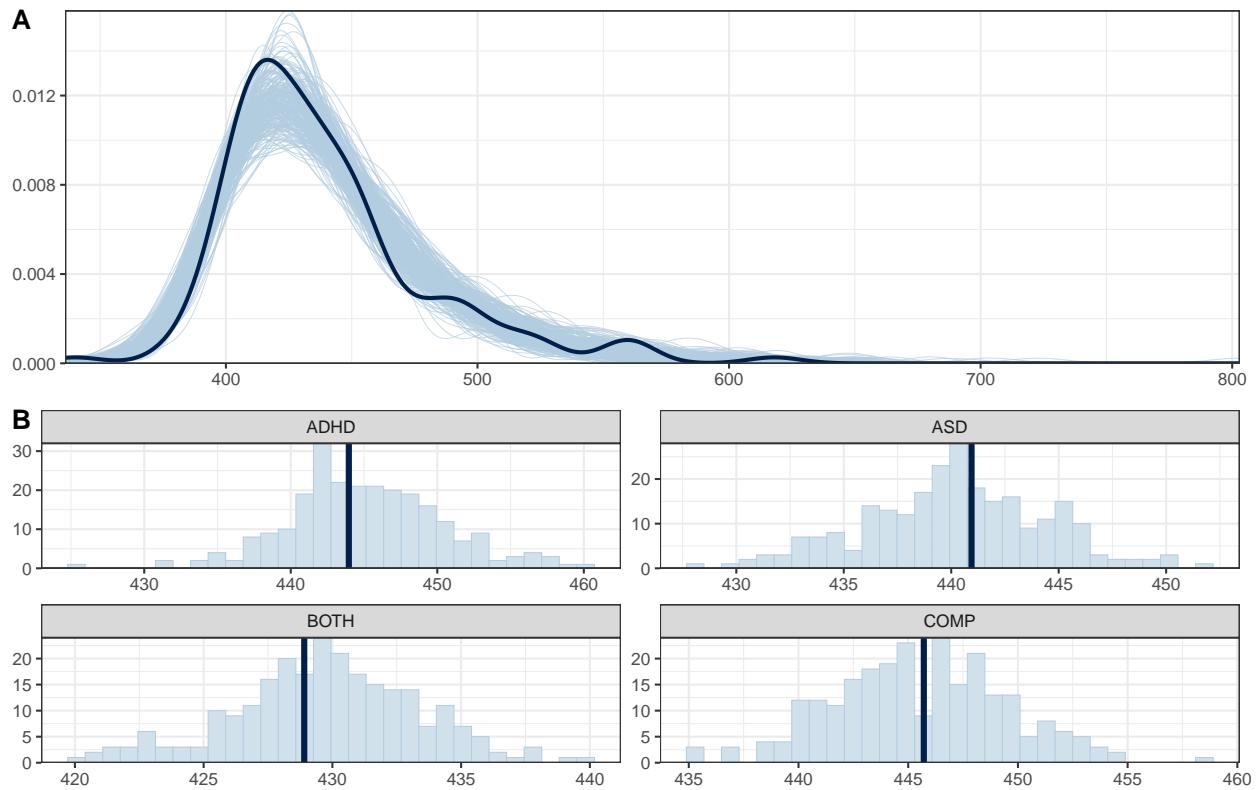
```
# get the posterior predictions
post.pred = posterior_predict(m.lat, ndraws = nsim)

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

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

Posterior predictive checks: latency



This looks much better with the simulated data based on the model capturing our actual data well.

Inferences

Now that we are convinced that we can trust our model, we have a look at the model and its estimates.

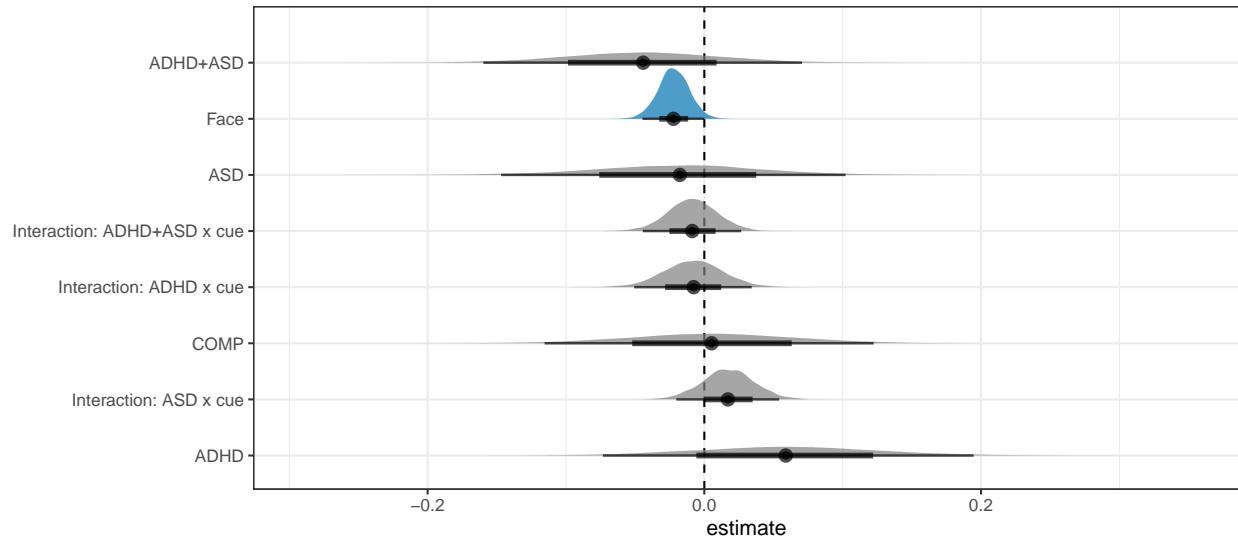
```
# print a summary
summary(m.lat)

## Family: shifted_lognormal
## Links: mu = identity; sigma = identity; ndt = identity
## Formula: lat ~ diagnosis * cue + (1 | subID)
## Data: df.lat.agg (Number of observations: 144)
## Draws: 4 chains, each with iter = 3000; warmup = 1000; thin = 1;
##        total post-warmup draws = 8000
##
## Multilevel Hyperparameters:
## ~subID (Number of levels: 76)
##             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)    0.30      0.04     0.23     0.39 1.00     1606     2660
## 
## Regression Coefficients:
##             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept       4.82      0.09     4.65     5.01 1.00     1883     2993
## diagnosis1      0.06      0.07    -0.07     0.19 1.00     1046     1940
## diagnosis2     -0.02      0.06    -0.15     0.10 1.00     1153     1948
## diagnosis3     -0.04      0.06    -0.16     0.07 1.00     1042     2027
## cue1           -0.02      0.01    -0.04    -0.00 1.00      7366     5787
```

```

## diagnosis1:cue1    -0.01      0.02     -0.05      0.03 1.00      5985      5124
## diagnosis2:cue1     0.02      0.02     -0.02      0.05 1.00      7074      5703
## diagnosis3:cue1    -0.01      0.02     -0.04      0.03 1.00      6622      5519
##
## Further Distributional Parameters:
##           Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma      0.13      0.02     0.10     0.16 1.00      2514      3608
## ndt       309.30     10.65   284.58   325.59 1.00      2173      2914
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
# plot the posterior distributions
as_draws_df(m.lat) %>%
  select(starts_with("b_")) %>%
  mutate(
    b_COMP     = - b_diagnosis1 - b_diagnosis2 - b_diagnosis3
  ) %>%
  pivot_longer(cols = starts_with("b_"), names_to = "coef", values_to = "estimate") %>%
  filter(coef != "b_Intercept") %>%
  mutate(
    coef = case_match(coef,
      "b_cue1" ~ "Face",
      "b_diagnosis1" ~ "ADHD",
      "b_diagnosis2" ~ "ASD",
      "b_diagnosis3" ~ "ADHD+ASD",
      "b_COMP" ~ "COMP",
      "b_diagnosis1:cue1" ~ "Interaction: ADHD x cue",
      "b_diagnosis2:cue1" ~ "Interaction: ASD x cue",
      "b_diagnosis3:cue1" ~ "Interaction: ADHD+ASD x cue"
    ),
    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, not_credible = c_light)) +
  theme(legend.position = "none")

```



```

# H2b: ASD(face) > COMP(face)
h2b = hypothesis(m.lat,
                  "0 < diagnosis1 + diagnosis3 + 2*diagnosis2 + diagnosis1:cue1 + diagnosis3:cue1 + 2*di
h2b

## Hypothesis Tests for class b:
##          Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(diagnosis1+d... < 0     0.01      0.1    -0.17     0.18      0.9
## Post.Prob Star
## 1       0.47
## ---
## 'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 95%;
## for two-sided hypotheses, the value tested against lies outside the 95%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.

# explore: faster towards faces
e = hypothesis(m.lat, "0 > cue1", alpha = 0.025)
e

## Hypothesis Tests for class b:
##          Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star
## 1 (0)-(cue1) > 0     0.02      0.01      0     0.04     39.4     0.98   *
## ---
## '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.lat %>%
  select(diagnosis, cue) %>%
  distinct() %>%
  mutate(
    condition = paste(diagnosis, cue, sep = "_")
  )
df.ms = as.data.frame(
  fitted(m.lat, summary = F,

```

```

    newdata = df.new %>% select(diagnosis, cue),
    re_formula = NA))
colnames(df.ms) = df.new$condition

# calculate our difference columns
df.ms = df.ms %>%
  mutate(
    COMP = (COMP_face + COMP_object)/2,
    ADHD = (ADHD_face + ADHD_object)/2,
    ASD = (ASD_object + ASD_face)/2,
    h2b = COMP_face - COMP_object,
    e = (ADHD_face + ASD_face + COMP_face)/3 - (ADHD_object + ASD_object + COMP_object)/3
  )

```

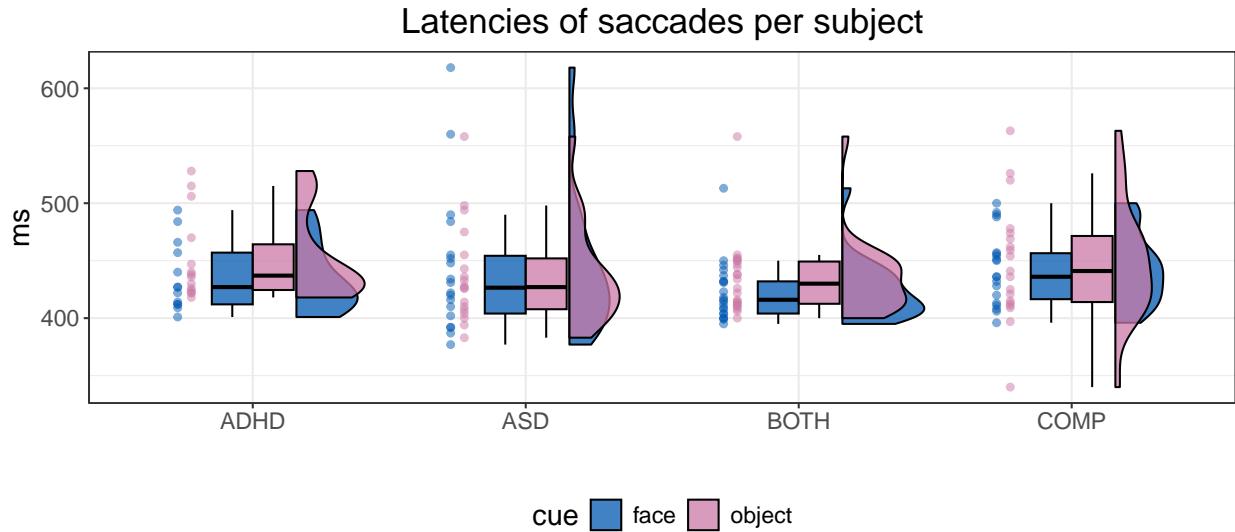
None of the population-level predictors showed credible effects on saccade latency. Specifically, we assessed whether latencies associated with saccades towards targets appearing at the previous location of the face are shorter in our comparison group than in the autistic group which was not the case (CI of ASD(face) - COMP(face): -16.34 to 5.11ms, posterior probability = 47.24%). We also explored whether our subjects produced faster saccades towards the targets appearing on the side of the face which also was not the case (CI of face - object: -11.77 to 1.89ms, posterior probability = 97.52%).

Plots

```

# rain cloud plot for the
df.lat.agg %>%
  ggplot(aes(diagnosis, lat, fill = cue, colour = cue)) +
  geom_rain(rain.side = 'r',
  boxplot.args = list(color = "black", outlier.shape = NA, show_guide = FALSE, alpha = 0.75),
  violin.args = list(color = "black", outlier.shape = NA, alpha = 0.75),
  boxplot.args.pos = list(
    position = ggpp::position_dodgegenudge(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_dodgegenudge(x = -0.25, width = 0.1))) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  labs(title = "Latencies of saccades per subject", x = "", y = "ms") +
  theme_bw() +
  theme(legend.position = "bottom",
    plot.title = element_text(hjust = 0.5),
    legend.direction = "horizontal",
    text = element_text(size = 15))

```



Bayes factor analysis

To complement our hypothesis testing using `brms::hypothesis()`, we perform a Bayes Factor analysis with models excluding some of our population-level predictors.

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

# 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(5599)
  m.lat.bf = brm(f.lat,
    df.lat.agg, prior = priors,
    iter = 40000, warmup = 10000,
    backend = "cmdstanr", threads = threading(8),
    family = "shifted_lognormal",
    file = "m_lat_agg_bf",
    save_pars = save_pars(all = TRUE)
  )
}
```

```

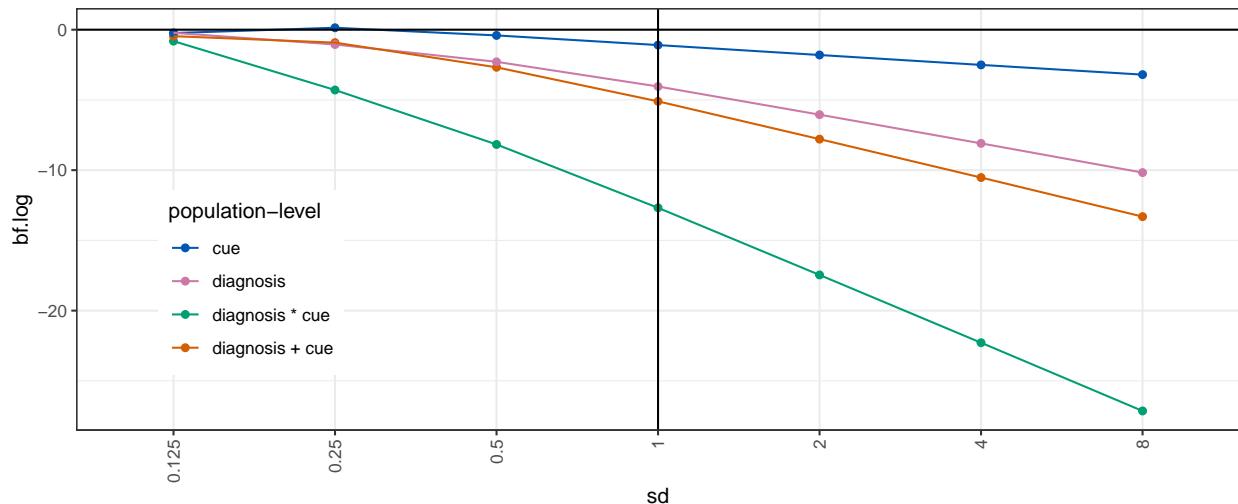
# loop through them
for (pr.desc in pr.descriptions) {
  tryCatch({
    # use the function
    bf_sens_2int(m.lat.bf, "diagnosis", "cue", pr.desc,
                 main.code, # prefix for all models and MLL
                 file.path(log_dir, "log_FAB_bf.txt"), # log file
                 sense_dir, # where to save the models and MLL
                 reps = 5
    )
  },
  error = function(err) {
    message(sprintf("Error for %s: %s", pr.desc, err))
  }
)
}

# read in the results
df.lat.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.lat.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") +
    #facet_wrap(. ~ `population-level`, scales = "free_y") +
    scale_colour_manual(values = custom.col) +
    theme_bw() +
    theme(legend.position = c(0.15,0.35),
          axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))

```

Sensitivity analysis with the intercept-only model as reference



```
# print the BFs based on chosen priors
kable(df.lat.bf %>% filter(priors == "chosen") %>% select(-priors) %>%
  filter(`population-level` != "1") %>% arrange(desc(bf.log)), digits = 3)
```

population-level	bf.log
cue	-1.094
diagnosis	-4.043
diagnosis + cue	-5.101
diagnosis * cue	-12.680

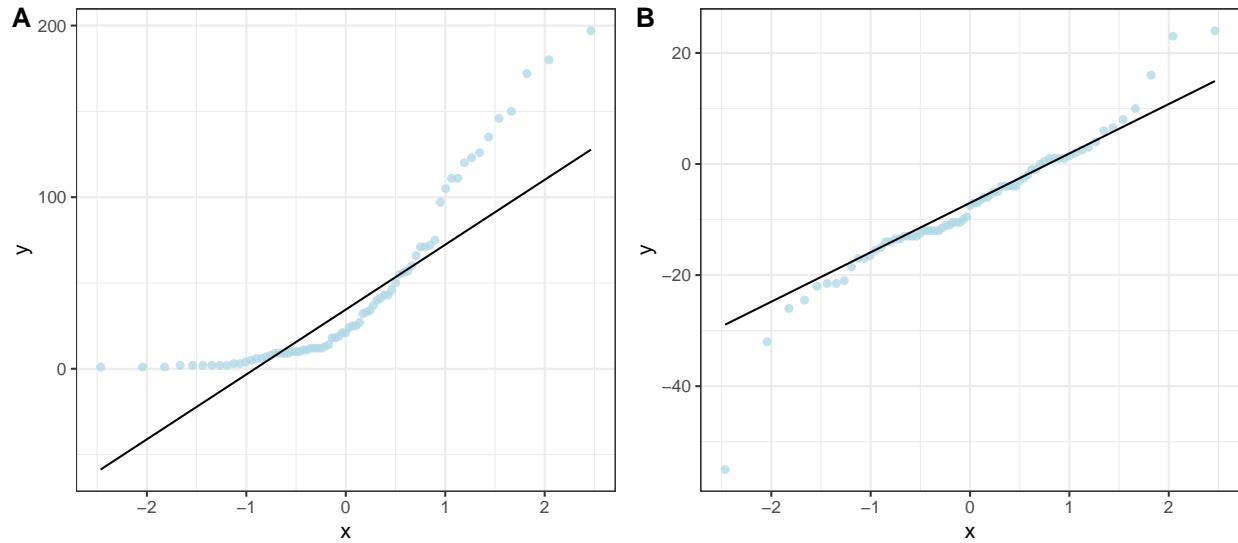
The null effects were also reflected in the Bayes Factor analysis where the intercept-only model consistently outperformed all other models (diagnostic groups: $\log(BF) = -4.043$; cue: $\log(BF) = -1.094$; diagnostic group and cue: $\log(BF) = -5.101$; full model: $\log(BF) = -12.68$).

S2.4 Correlation with reaction times: number of saccades

Last, we hypothesised that the FAB effect on reaction times may be associated with saccades produced towards the face. To investigate this, we use a Bayesian Spearman correlation as both FAB effect and number of saccades are not normally distributed.

```
# only keep saccades towards faces
df.diff = df.cnt %>% filter(dir_face == TRUE)

# check the distribution plot > not normally distributed
p1 = ggplot(df.diff, aes(sample = n.face)) +
  stat_qq(alpha = 0.75, colour = "lightblue") +
  stat_qq_line() +
  theme_bw()
p2 = ggplot(df.diff, aes(sample = rt.fab)) +
  stat_qq(alpha = 0.75, colour = "lightblue") +
  stat_qq_line() +
  theme_bw()
ggarrange(p1, p2,
           nrow = 1, ncol = 2, labels = "AUTO")
```



```

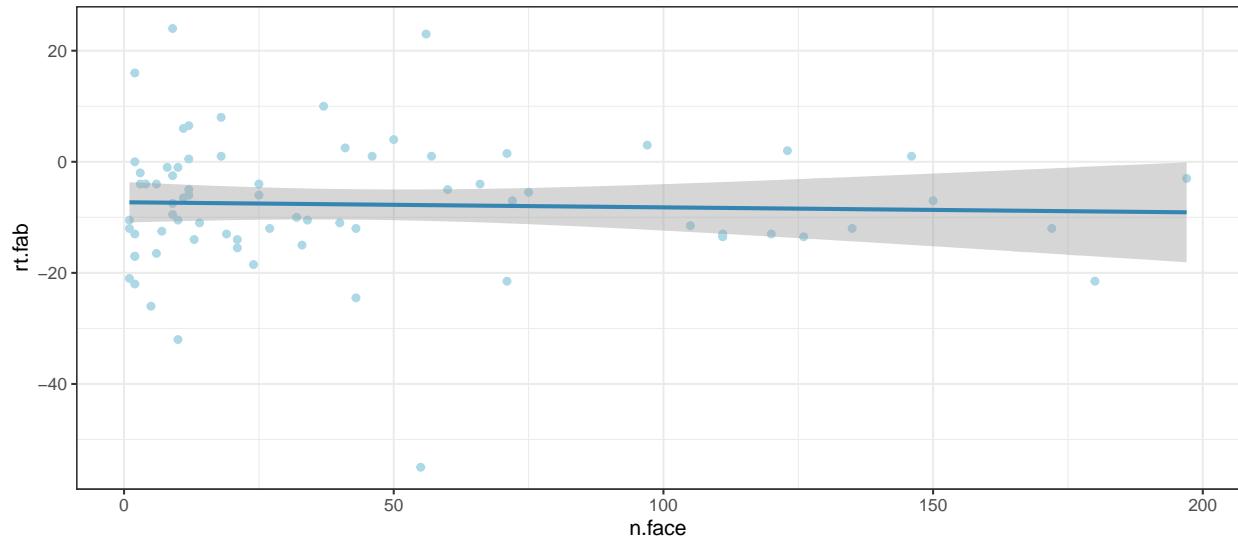
# do a Bayesian Spearman correlation: https://osf.io/j5wud
source("./helpers/rankBasedCommonFunctions.R")
source("./helpers/spearmanSampler.R")

# Default beta prior width is set to a = b = 1 for the sampler
if (file.exists("CNT_rho.rds")) {
  rhoSamples.cnt = readRDS("CNT_rho.rds")
} else {
  rhoSamples.cnt =
    spearmanGibbsSampler(xVals = df.diff$n.face,
                          yVals = df.diff$rt.fab,
                          nSamples = 5e3)
  saveRDS(rhoSamples.cnt, file = "CNT_rho.rds")
}

# give the posterior samples for rho to the function below to compute BF01
cor.bf = computeBayesFactorOneZero(rhoSamples.cnt$rhoSamples,
                                    whichTest = "Spearman",
                                    priorParameter = 1)

# visualise it
ggplot(data = df.diff, aes(x = n.face, y = rt.fab)) +
  geom_point(colour = "lightblue") +
  geom_smooth(method = "lm",
              formula = y ~ x,
              geom = "smooth", colour = c_mid_highlight) +
  theme_bw()

```



Furthermore, we assessed the relationship between face attention bias and number of saccades produced towards the face on the participant level (see supplementary materials S2.4). We used a Bayesian Spearman correlation due to both values not being normally distributed. This model revealed no association between number of saccades and face attention bias, in fact there was moderate evidence against an association between the number of saccades and face attention bias ($\log(BF) = -1.898$).

S2.5 Exploration: number of saccades towards cue

Additionally to our hypotheses, we also explored any effects of diagnostic status, cue type and their interaction on the number of saccades during the presentation of the cues.

Specify the model

```
code = "CNT-cue"

# set the formula
f.cnt = brms::bf(n.cue ~ diagnosis * dir_face + (1 | subID))

# set priors based on study design
priors = c(
  prior(normal(3, 1.5), class = Intercept),
  prior(normal(0, 1.0), class = sd),
  prior(normal(0, 1.0), class = b)
)

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

# 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
  set.seed(2468)
  gen = SBC_generator_brms(f.cnt, data = df.cnt.cue, prior = priors,
    thin = 50, warmup = 10000, refresh = 2000,
    generate_lp = TRUE, family = poisson(), init = 0.1)
  bck = SBC_backend_brms_from_generator(gen, chains = 4, thin = 1,
    warmup = warm, iter = iter)
  dat = generate_datasets(gen, nsim)
  saveRDS(dat, file.path(cache_dir, sprintf("dat_%s.rds", code)))
  res = compute_SBC(dat,
    bck,
    cache_mode      = "results",
    cache_location = file.path(cache_dir, sprintf("res_%s", code)))
  saveRDS(res$stats,
    file = file.path(cache_dir, paste0("df_res_", code, ".rds")))
  saveRDS(res$backend_diagnostics,
    file = file.path(cache_dir, paste0("df_div_", code, ".rds")))
}

```

We start by investigating the rhats and the number of divergent samples. This shows that 5 of 250 simulations had at least one parameter that had an rhat of at least 1.05, and only 0 models had divergent samples. This suggests that this model performs well.

Prior predictive checks

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

```

# get the true values
truePars = dat[["variables"]]

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

# set very large data points to a value of 432
dvfakematH = dvfakemat;
dvfakematH[dvfakematH > 432] = 432
# compute one histogram per simulated data-set
breaks = seq(0, max(dvfakematH, na.rm=T), length.out = 100)
binwidth = round(breaks[2] - breaks[1])
breaks = seq(0, max(dvfakematH, na.rm=T), binwidth)
histmat = matrix(NA, ncol = nrow(truePars) + binwidth, nrow = length(breaks)-1)
for (i in 1:nrow(truePars)) {
  histmat[,i] = hist(dvfakematH[,i], breaks = breaks, plot = F)$counts
}
# for each bin, compute quantiles across histograms
probs = seq(0.1, 0.9, 0.1)
quantmat= as.data.frame(matrix(NA, nrow=dim(histmat)[1], ncol = length(probs)))
names(quantmat) = paste0("p", probs)
for (i in 1:dim(histmat)[1]) {

```

```

quantmat[i,] = quantile(histmat[i,], p = probs, na.rm = T)
}
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 = "Prior predictive distribution", y = "", x = "number of saccades") +
  theme_bw()

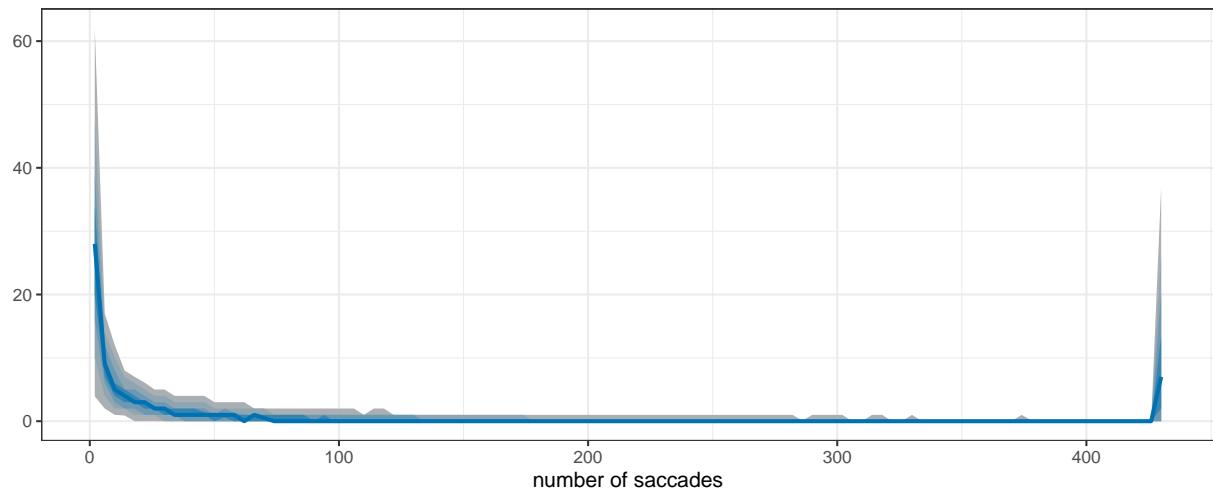
tmpM = apply(dvfakemathH, 2, mean) # mean
tmpSD = apply(dvfakemathH, 2, sd)
p2 = ggplot() +
  stat_bin(aes(x = tmpM), fill = c_dark) +
  labs(x = "Mean number of saccades", title = "Means of simulated data") +
  theme_bw()
p3 = ggplot() +
  stat_bin(aes(x = tmpSD), fill = c_dark) +
  labs(x = "SD number of saccades", title = "Standard deviations of simulated data") +
  theme_bw()

p = ggarrange(p1,
  ggarrange(p2, p3, ncol = 2, labels = c("B", "C")),
  nrow = 2, labels = "A")
annotate_figure(p, top = text_grob("Prior predictive checks", face = "bold", size = 14))

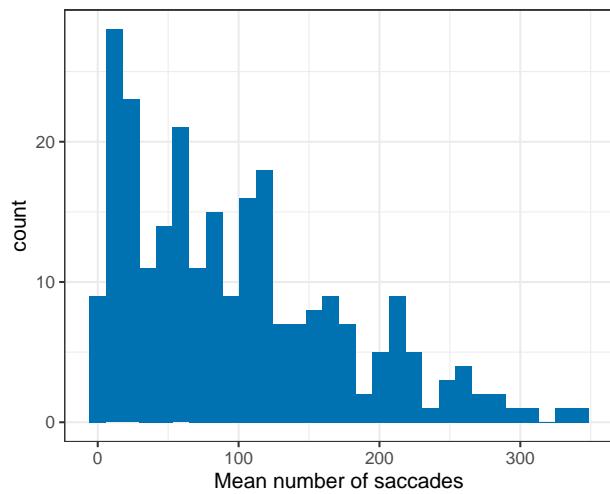
```

Prior predictive checks

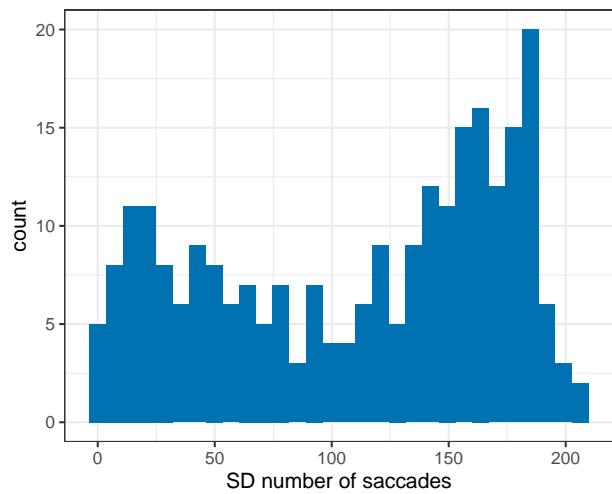
A Prior predictive distribution



B Means of simulated data



C Standard deviations of simulated data



Computational faithfulness and model sensitivity

```
# get simulation numbers with issues
des_rank = max(df.results$max_rank)
check = merge(df.results %>%
  group_by(sim_id) %>%
  summarise(
    rhat = max(rhat, na.rm = T),
    mean_rank = max(max_rank)
  ) %>%
  filter(rhat >= 1.05 | mean_rank < des_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)) %>%
```

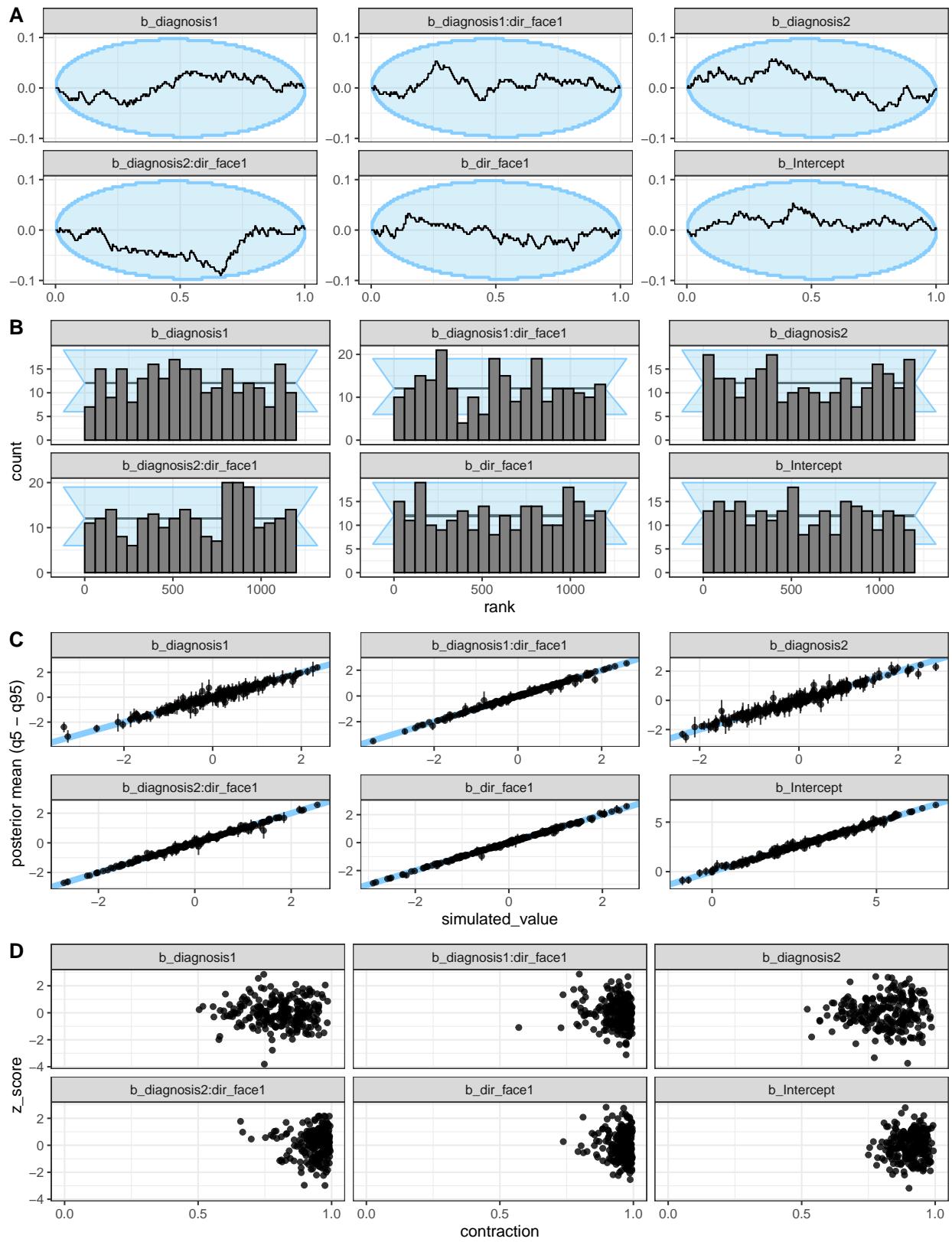
```

ungroup() %>%
  mutate(
    max_rank = max(rank)
  )
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 = 0.75) + 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)),
      rep(
        as.numeric(
          gsub(".*, (.+)\\".*", "\\\\"1",
            priors[priors$class == "b",]$prior)),
        length(unique(df.results.b$variable))-1)),
      unique(df.results.b$variable))) +
  theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))

p = ggarrange(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))

```

Computational faithfulness and model sensitivity



Second, 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.

Third, 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). There seem to be singular models where the posterior sd was still larger than the prior sd. However, since we already have quite wide priors, especially for the Intercept, we go ahead with this model.

Posterior predictive checks

As the next step, we fit the model and check whether the chains have converged, which they seem to have. We then perform posterior predictive checks on the model using the bayesplot package.

```
# fit the model
set.seed(6644)
m.cnt = brm(f.cnt,
             df.cnt.cue, prior = priors,
             iter = iter, warmup = warm,
             backend = "cmdstanr", threads = threading(8),
             file = "m_cnt-cue",
             family = "poisson",
             save_pars = save_pars(all = TRUE)
            )
rstan::check_hmc_diagnostics(m.cnt$fit)

##
## Divergences:
## 0 of 12000 iterations ended with a divergence.

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

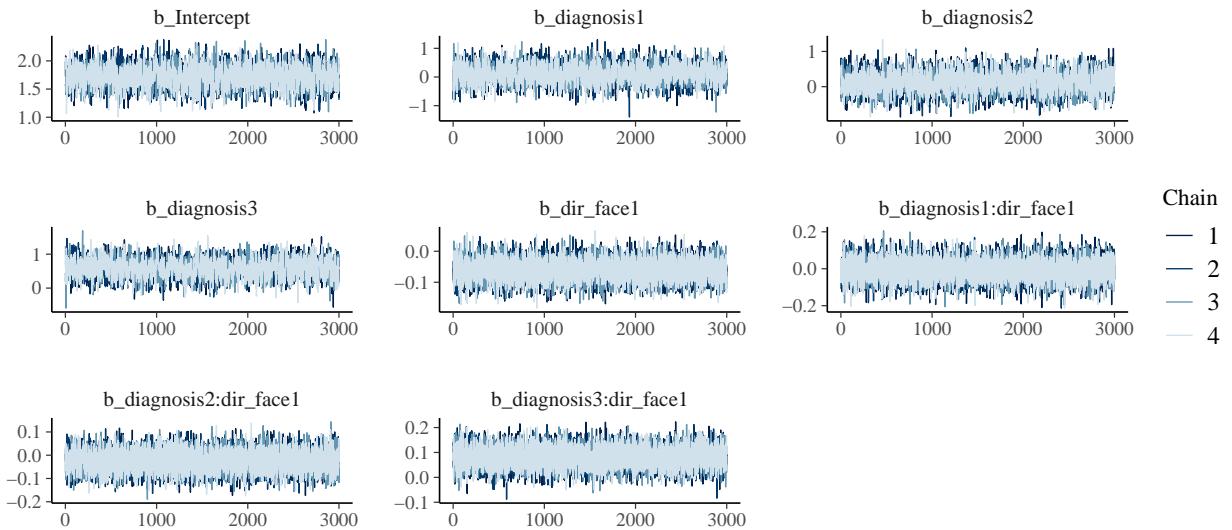
##
## Energy:
## E-BFMI indicated no pathological behavior.

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

## [1] 0

# check the trace plots
post.draws = as_draws_df(m.cnt)
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 divergent samples and no rhats that are higher or equal to 1.01. Therefore, we go ahead and perform our posterior predictive checks.

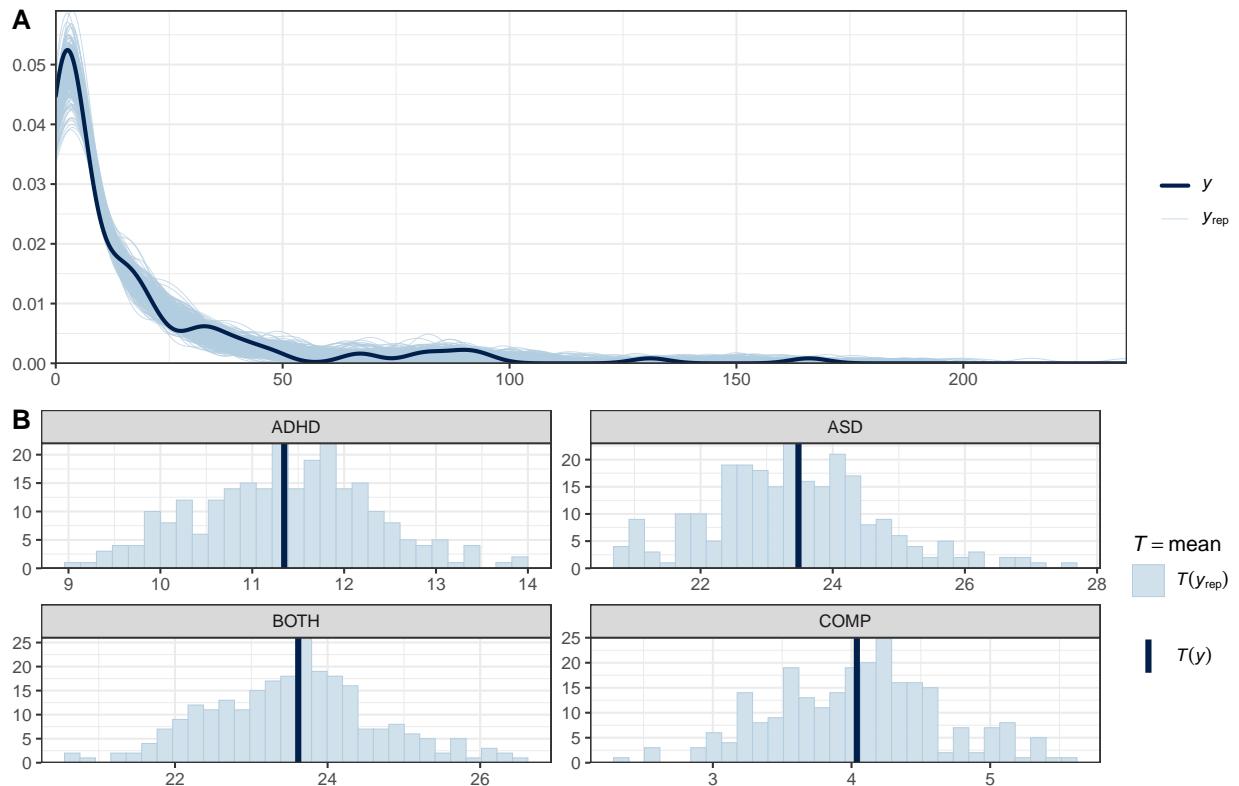
```
# get the posterior predictions
post.pred = posterior_predict(m.cnt, ndraws = nsim)

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

# distributions of means and sds compared to the real values per group
p2 = ppc_stat_grouped(df.cnt.cue$n.cue, post.pred, df.cnt.cue$diagnosis) +
  theme_bw()

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

Posterior predictive checks: number of saccades towards cue



The predictions based on the model capture the data well. This further increased our trust in the model and we move on to interpret its parameter.

Inferences

Now that we are convinced that we can trust our model, we have a look at the model and its estimates.

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

## Family: poisson
## Links: mu = log
## Formula: n.cue ~ diagnosis * dir_face + (1 | subID)
## Data: df.cnt.cue (Number of observations: 115)
## Draws: 4 chains, each with iter = 4500; warmup = 1500; thin = 1;
##         total post-warmup draws = 12000
##
## Multilevel Hyperparameters:
## ~subID (Number of levels: 64)
##             Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)    1.30     0.13    1.07    1.59 1.00      2742     4947
## 
## Regression Coefficients:
##                         Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                1.73     0.18    1.37    2.08 1.00      2546
## diagnosis1                 0.11     0.31   -0.50    0.72 1.00      2435
## diagnosis2                 0.14     0.28   -0.42    0.68 1.00      2466
## diagnosis3                 0.58     0.26    0.07    1.10 1.00      1907
```

```

## dir_face1      -0.06    0.03   -0.12    0.00 1.00    10629
## diagnosis1:dir_face1 -0.01    0.06   -0.12    0.10 1.00    21186
## diagnosis2:dir_face1 -0.02    0.04   -0.11    0.06 1.00    13804
## diagnosis3:dir_face1  0.08    0.04    0.00    0.16 1.00    13098
##                                     Tail_ESS
## Intercept                  4551
## diagnosis1                 4276
## diagnosis2                 4867
## diagnosis3                 3785
## dir_face1                  9014
## diagnosis1:dir_face1     8065
## diagnosis2:dir_face1     9655
## diagnosis3:dir_face1     9501
##
## 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.cnt = as_draws_df(m.cnt) %>%
  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,
    b_dir_faceTRUE = - b_dir_face1,
    COMP        = b_Intercept + b_COMP
  )

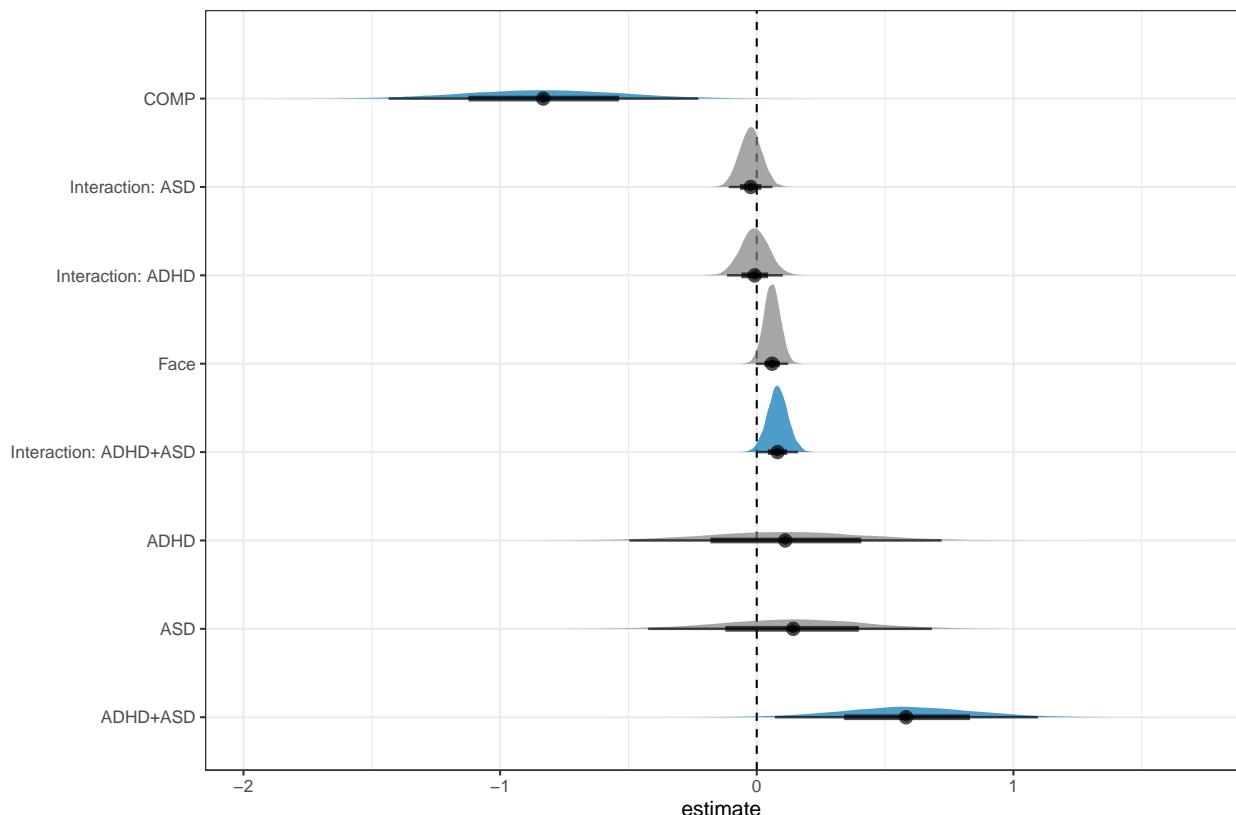
# plot the posterior distributions
df.m.cnt %>%
  select(starts_with("b_")) %>%
  pivot_longer(cols = starts_with("b_"), names_to = "coef", values_to = "estimate") %>%
  filter(coef != "b_Intercept" & coef != "b_dir_face1") %>%
  mutate(
    coef = case_match(coef,
      "b_diagnosis1" ~ "ADHD",
      "b_diagnosis2" ~ "ASD",
      "b_diagnosis3" ~ "ADHD+ASD",
      "b_COMP"       ~ "COMP",
      "b_dir_faceTRUE" ~ "Face",
      "b_diagnosis1:dir_face1" ~ "Interaction: ADHD",
      "b_diagnosis3:dir_face1" ~ "Interaction: ADHD+ASD",
      "b_diagnosis2:dir_face1" ~ "Interaction: 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")

```



```

# exploration: COMP: face > object
hypothesis(m.cnt, "0 > dir_face1 - diagnosis1:dir_face1 - diagnosis2:dir_face1 - diagnosis3:dir_face1",
           alpha = 0.025)

## Hypothesis Tests for class b:
##                               Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(dir_face1-di... > 0      0.11       0.1     -0.09      0.3      6.1
##   Post.Prob Star
## 1      0.86
## ---
## '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: face > object
e = hypothesis(m.cnt, "0 > dir_face1", alpha = 0.025)
e

```

```

## Hypothesis Tests for class b:
##                               Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob
## 1 (0)-(dir_face1) > 0      0.06       0.03      0     0.12      30.33      0.97
##   Star
## 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.
# exploration: ADHD > COMP
hypothesis(m.cnt, "0 < 2*diagnosis1 + diagnosis2 + diagnosis3", alpha = 0.025)

## Hypothesis Tests for class b:
##                               Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis1... < 0    -0.94       0.52     -1.98      0.08      27.71
##   Post.Prob Star
## 1      0.97
## ---
## '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: ADHD(face) > ADHD(object)
hypothesis(m.cnt, "0 > dir_face1 + diagnosis1:dir_face1", alpha = 0.025)

## Hypothesis Tests for class b:
##                               Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(dir_face1+di... > 0      0.07       0.06     -0.06      0.19      6.21
##   Post.Prob Star
## 1      0.86
## ---
## '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: ASD > COMP
hypothesis(m.cnt, "0 < 2*diagnosis2 + diagnosis1 + diagnosis3", alpha = 0.025)

## Hypothesis Tests for class b:
##                               Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis2... < 0    -0.97       0.48     -1.91     -0.02      43.94
##   Post.Prob Star
## 1      0.98      *
## ---
## '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: BOTH > COMP
hypothesis(m.cnt, "0 < diagnosis2 + diagnosis1 + 2*diagnosis3", alpha = 0.025)

## Hypothesis Tests for class b:

```

```

## Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(diagnosis2+d... < 0     -1.42      0.46    -2.31     -0.5     630.58
##   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.
# exploration: ASD(face) > ASD(object)
hypothesis(m.cnt, "0 > dir_face1 + diagnosis2:dir_face1", alpha = 0.025)

## Hypothesis Tests for class b:
## Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(dir_face1+di... > 0     0.08      0.04      0     0.16     53.3
##   Post.Prob Star
## 1          0.98      *
## ---
## '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: BOTH(face) > BOTH(object)
hypothesis(m.cnt, "0 > dir_face1 + diagnosis3:dir_face1", alpha = 0.025)

## Hypothesis Tests for class b:
## Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(dir_face1+di... > 0     -0.02      0.03    -0.09     0.04     0.36
##   Post.Prob Star
## 1          0.26
## ---
## '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.cnt.cue %>%
  select(diagnosis, dir_face) %>%
  distinct() %>%
  mutate(
    condition = paste(diagnosis, dir_face, sep = "_")
  )
df.est = as.data.frame(
  fitted(m.cnt, summary = F,
         newdata = df.new %>% select(diagnosis, dir_face),
         re_formula = NA))
colnames(df.est) = df.new$condition

```

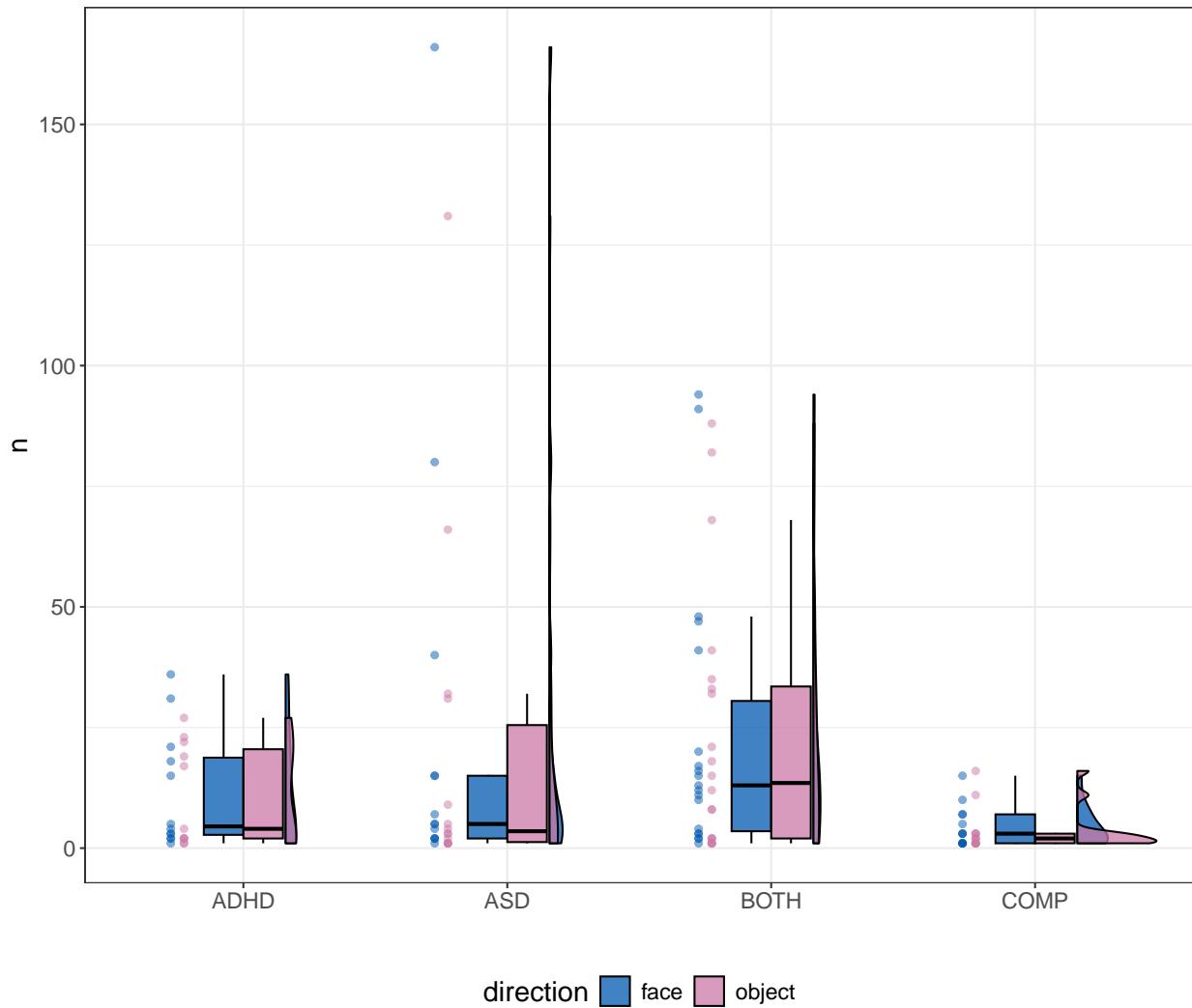
[!ADD]

Plots

```
# rain cloud plot

df.cnt.cue %>%
  mutate(direction = if_else(dir_face == T, "face", "object")) %>%
  ggplot(aes(diagnosis, n.cue, fill = direction, colour = direction)) +
  geom_rain(rain.side = 'r',
  boxplot.args = list(color = "black", outlier.shape = NA, show_guide = FALSE, alpha = 0.75),
  violin.args = list(color = "black", outlier.shape = NA, alpha = 0.75),
  boxplot.args.pos = list(
    position = ggpp::position_dodge_nudge(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_dodge_nudge(x = -0.25, width = 0.1))) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  labs(title = "Number of saccades per subject", x = "", y = "n") +
  theme_bw() +
  theme(legend.position = "bottom",
    plot.title = element_text(hjust = 0.5),
    legend.direction = "horizontal",
    text = element_text(size = 15))
```

Number of saccades per subject



Bayes factor analysis

To complement our hypothesis testing using `brms::hypothesis()`, we perform a Bayes Factor analysis with models excluding some of our population-level predictors.

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

# 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(3311)
  m.cnt.bf = brm(f.cnt,
    df.cnt.cue, prior = priors,
    iter = 40000, warmup = 10000,
    backend = "cmdstanr", threads = threading(8),
    file = "m_cnt-cue_bf",
    family = "poisson",
    save_pars = save_pars(all = TRUE)
  )
}

# loop through them
for (pr.desc in pr.descriptions) {
  tryCatch({
    # use the function
    bf_sens_2int(m.cnt.bf, "diagnosis", "dir_face", pr.desc,
      main.code, # prefix for all models and MLL
      file.path(log_dir, "log_FAB_bf.txt"), # log file
      sense_dir, # where to save the models and MLL
      reps = 5
    )
  },
  error = function(err) {
    message(sprintf("Error for %s: %s", pr.desc, err))
  }
)
}

# read in the results
df.cnt.bf = read_csv(file.path(sense_dir, sprintf("df_%s_bf.csv", main.code)),
  show_col_types = F)

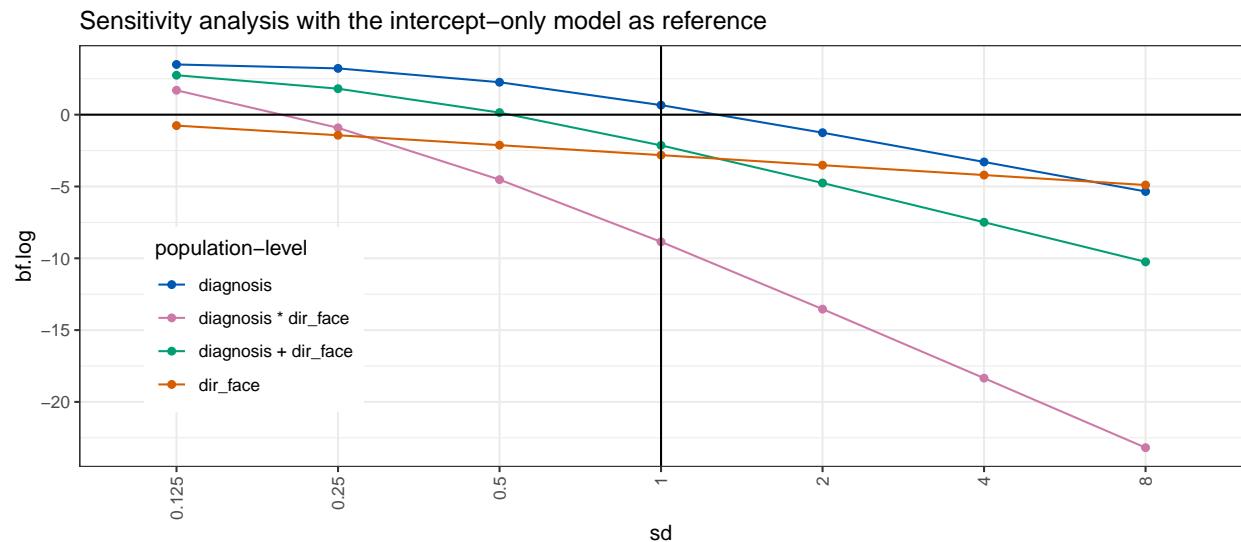
# check the sensitivity analysis result per model
df.cnt.bf %>%
  filter(`population-level` != "1") %>%
  mutate(
    sd = as.factor(case_when(
      priors == "chosen" ~ "1",
      substr(priors, 1, 3) == "sdx" ~ gsub("sdx", "", priors),
      T ~ priors
    )),
    order = case_when(
      priors == "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") +
#facet_wrap(. ~ `population-level`, scales = "free_y") +
scale_colour_manual(values = custom.col) +
theme_bw() +
theme(legend.position = c(0.15,0.35),
      axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))

```



```

# let's focus on a smaller range of bayes factor values to see the differences better
df.cnt.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`)) +

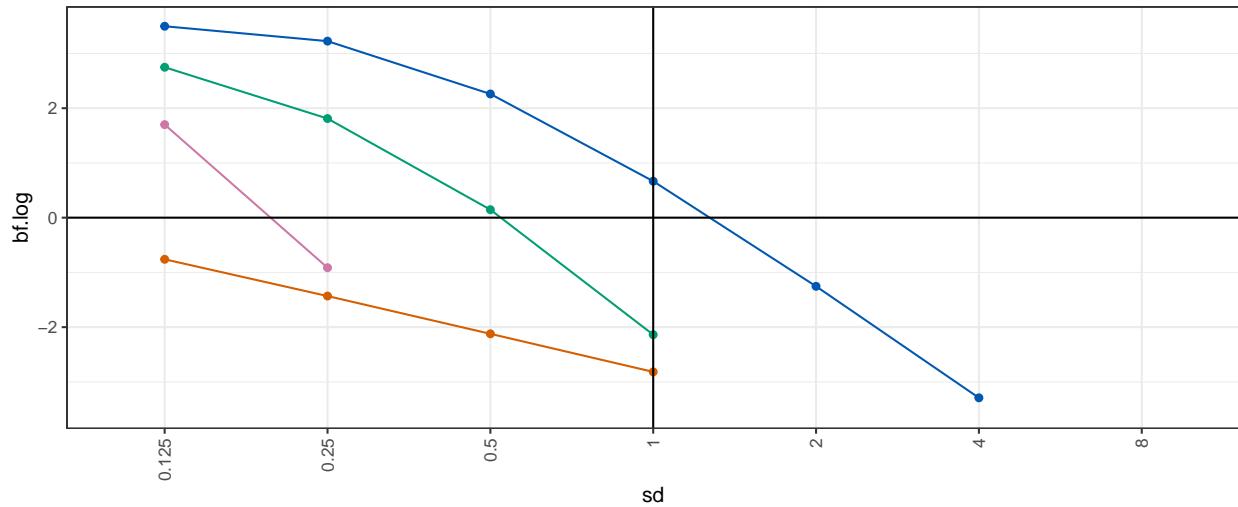
```

```

geom_point() +
geom_line() +
geom_vline(xintercept = "1") +
geom_hline(yintercept = 0) +
ggtitle("Sensitivity analysis with the intercept-only model as reference") +
ylim(-3.5, 3.5) +
scale_colour_manual(values = custom.col) +
theme_bw() +
theme(legend.position = "none",
      axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))

```

Sensitivity analysis with the intercept–only model as reference



```

# print the BFs based on chosen priors
kable(df.cnt.bf %>% filter(priors == "chosen") %>% select(-priors) %>%
  filter(`population-level` != "1") %>% arrange(desc(bf.log)), digits = 3)

```

population-level	bf.log
diagnosis	0.666
diagnosis + dir_face	-2.138
dir_face	-2.817
diagnosis * dir_face	-8.854

However, in the Bayes Factor analysis, the model including whether the saccade was produced towards the face or the object performs only slightly better than the intercept-only model, with strong evidence against the model including saccade direction (comparisons with intercept-only model, diagnostic groups: $\log(BF) = 0.666$; saccade direction: $\log(BF) = -2.817$; diagnostic group and saccade direction: $\log(BF) = -2.138$; full model: $\log(BF) = -8.854$). Additionally, the sensitivity analysis shows that this effect is rather inconsistent across priors which decreases our confidence in this result. Therefore, we conclude that more data is needed to determine whether consistently more saccades towards the faces are produced.

S2.6 Exploration: number of saccades towards target

Similarly, we explore the number of saccades during the target presentation and if the number of saccades differs depending on whether the target appears on the side of the face or the object cue.

Specify the model

```
code = "CNT-tar"

# set the formula
f.cnt = brms::bf(n.tar ~ diagnosis * cue + (1 | subID))

# set priors based on study design
priors = c(
  prior(normal(3, 1.5), class = Intercept),
  prior(normal(0, 1.0), class = sd),
  prior(normal(0, 1.0), class = b)
)

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

SBC

```
# 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
  set.seed(2468)
  gen = SBC_generator_brms(f.cnt, data = df.cnt.tar, prior = priors,
    thin = 50, warmup = 10000, refresh = 2000,
    generate_lp = TRUE, family = poisson(), init = 0.1)
  bck = SBC_backend_brms_from_generator(gen, chains = 4, thin = 1,
    warmup = warm, iter = iter)
  dat = generate_datasets(gen, nsim)
  saveRDS(dat, file.path(cache_dir, sprintf("dat_%s.rds", code)))
  res = compute_SBC(dat,
    bck,
    cache_mode     = "results",
    cache_location = file.path(cache_dir, sprintf("res_%s", code)))
  saveRDS(res$stats,
    file = file.path(cache_dir, paste0("df_res_", code, ".rds")))
  saveRDS(res$backend_diagnostics,
    file = file.path(cache_dir, paste0("df_div_", code, ".rds")))
}
```

We start by investigating the rhats and the number of divergent samples. This shows that 5 of 250 simulations had at least one parameter that had an rhat of at least 1.05, but 1 models had divergent samples. This suggests that this model performs well.

Computational faithfulness and model sensitivity

```
# get simulation numbers with issues
des_rank = max(df.results$max_rank)
```

```

check = merge(df.results %>%
  group_by(sim_id) %>%
  summarise(
    rhat = max(rhat, na.rm = T),
    mean_rank = max(max_rank)
  ) %>%
  filter(rhat >= 1.05 | mean_rank < des_rank),
  df.backend %>% filter(n_divergent > 0), all = T)

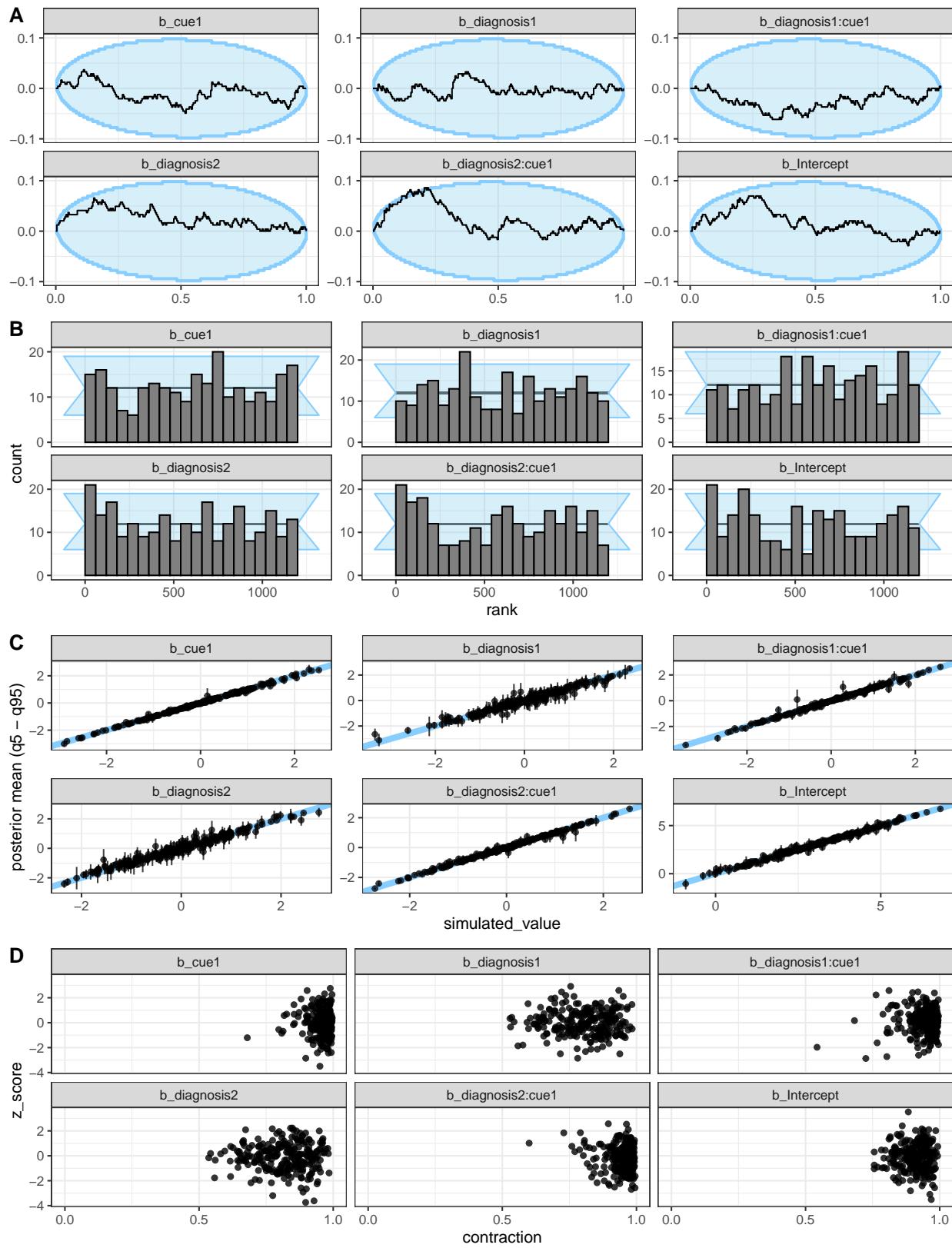
# plot SBC with functions from the SBC package focusing on population-level parameters

df.results.b = df.results %>%
  filter(substr(variable, 1, 2) == "b_") %>%
  filter(!(sim_id %in% check$sim_id)) %>%
  ungroup() %>%
  mutate(
    max_rank = max(rank)
  )
p1 = plot_ecdf_diff(df.results.b) + theme_bw() + theme(legend.position = "none") +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p2 = plot_rank_hist(df.results.b, bins = 20) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p3 = plot_sim_estimated(df.results.b, alpha = 0.75) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p4 = plot_contraction(df.results.b,
  prior_sd = setNames(
    c(as.numeric(
      gsub(".*, (.+)\\\".*", "\\"1",
        priors[priors$class == "Intercept",]$prior)),
    rep(
      as.numeric(
        gsub(".*, (.+)\\\".*", "\\"1",
          priors[priors$class == "b",]$prior)),
      length(unique(df.results.b$variable))-1)),
    unique(df.results.b$variable))) +
  theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))

p = ggarrange(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))

```

Computational faithfulness and model sensitivity



Second, 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.

Third, 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). There seem to be singular models where the posterior sd was still larger than the prior sd. However, since we already have quite wide priors, especially for the Intercept, we go ahead with this model.

Posterior predictive checks

As the next step, we fit the model and check whether the chains have converged, which they seem to have. We then perform posterior predictive checks on the model using the bayesplot package.

```
# fit the maximal model
set.seed(7744)
m.cnt.tar = brm(f.cnt,
                 df.cnt.tar, prior = priors,
                 iter = iter, warmup = warm,
                 backend = "cmdstanr", threads = threading(8),
                 file = "m_cnt-tar",
                 family = "poisson",
                 save_pars = save_pars(all = TRUE)
                 )
rstan::check_hmc_diagnostics(m.cnt$fit)

##
## Divergences:
## 0 of 12000 iterations ended with a divergence.

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

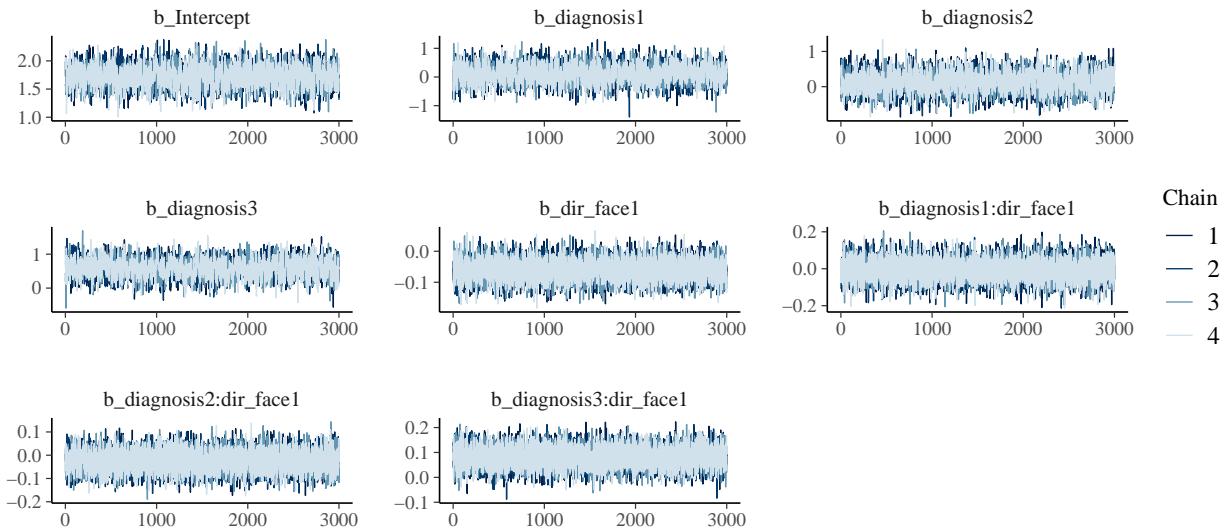
##
## Energy:
## E-BFMI indicated no pathological behavior.

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

## [1] 0

# check the trace plots
post.draws = as_draws_df(m.cnt)
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 divergent samples and no rhats that are higher or equal to 1.01. Therefore, we go ahead and perform our posterior predictive checks.

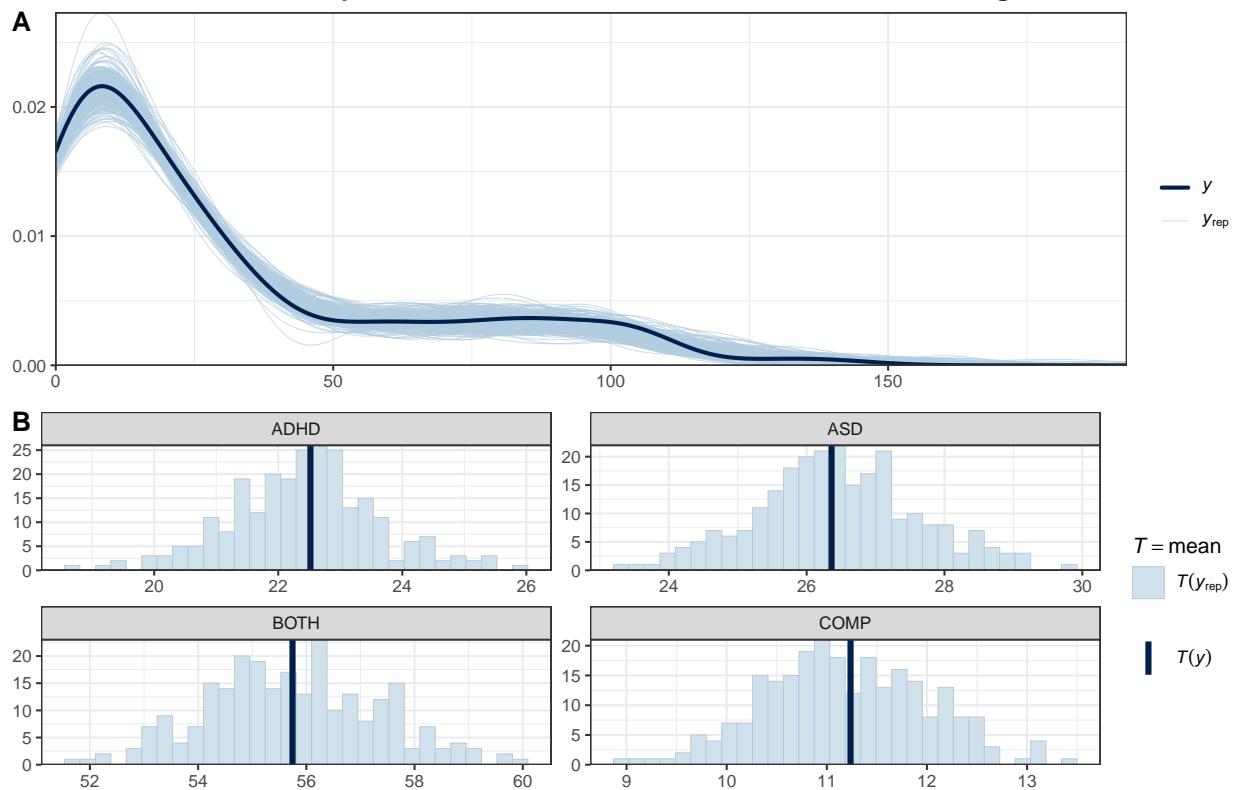
```
# get the posterior predictions
post.pred = posterior_predict(m.cnt.tar, ndraws = nsim)

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

# distributions of means and sds compared to the real values per group
p2 = ppc_stat_grouped(df.cnt.tar$n.tar, post.pred, df.cnt.tar$diagnosis) +
  theme_bw()

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

Posterior predictive checks: number of saccades towards target



The predictions based on the model capture the data well. This further increased our trust in the model and we move on to interpret its parameter.

Model summary

Now that we are convinced that we can trust our model, we have a look at the model and its estimates.

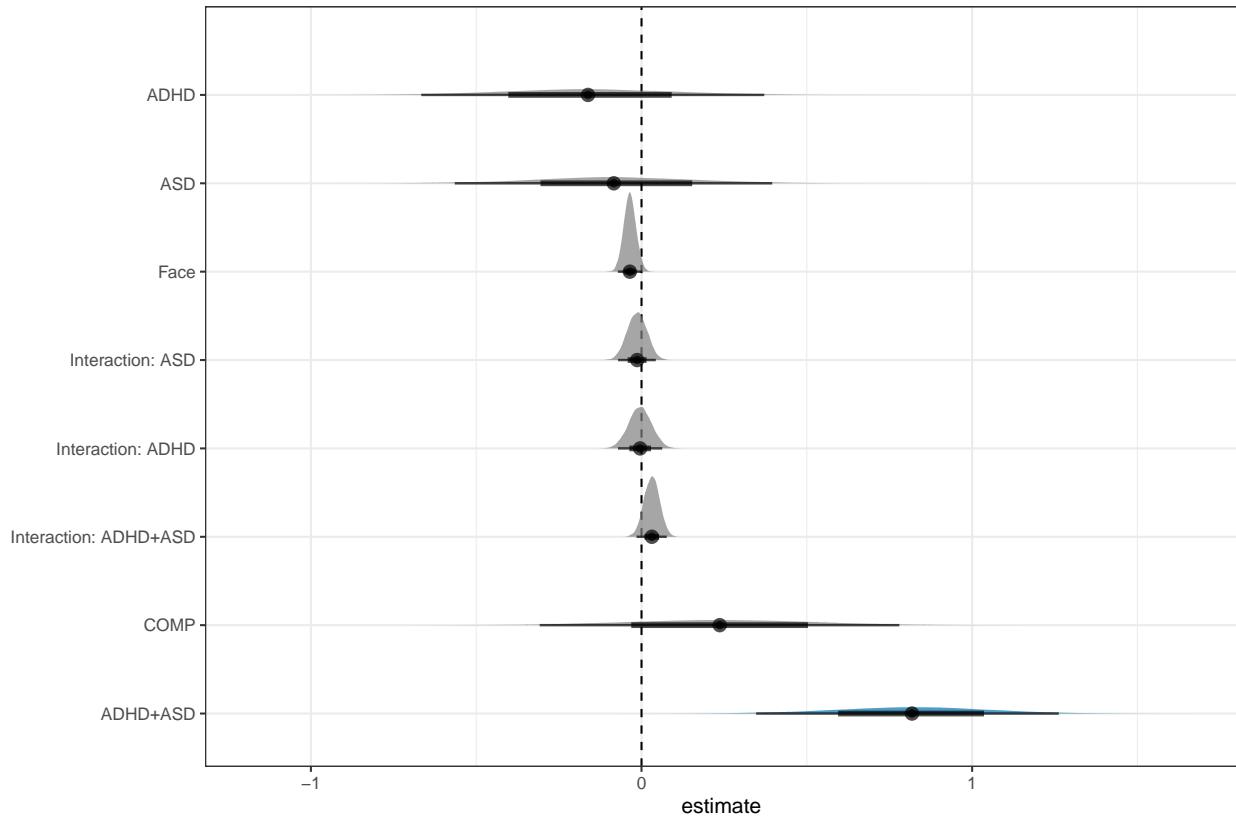
```
# print a summary
summary(m.cnt.tar)

## Family: poisson
## Links: mu = log
## Formula: n.tar ~ diagnosis * cue + (1 | subID)
## Data: df.cnt.tar (Number of observations: 144)
## Draws: 4 chains, each with iter = 4500; warmup = 1500; thin = 1;
##         total post-warmup draws = 12000
##
## Multilevel Hyperparameters:
## ~subID (Number of levels: 76)
##             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)    1.22      0.11     1.03     1.47 1.00     2356     4302
##
## Regression Coefficients:
##             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept        2.62      0.15     2.33     2.91 1.00     2146     3050
## diagnosis1      -0.16      0.26    -0.67     0.37 1.00     1879     3287
## diagnosis2      -0.08      0.24    -0.56     0.40 1.00     1953     2842
## diagnosis3       0.82      0.23     0.35     1.26 1.00     1645     2689
```

```

## cue1      -0.04    0.02   -0.07    0.00 1.00    12115    9440
## diagnosis1:cue1 -0.00    0.03   -0.07    0.06 1.00    13961    9242
## diagnosis2:cue1 -0.01    0.03   -0.07    0.04 1.00    13194    8965
## diagnosis3:cue1  0.03    0.02   -0.02    0.08 1.00    11665    9265
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
# plot the posterior distributions
as_draws_df(m.cnt.tar) %>%
  select(starts_with("b_")) %>%
  mutate(
    b_COMP     = - b_diagnosis1 - b_diagnosis2
  ) %>%
  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, not_credible)) +
  theme(legend.position = "none")

```



```
# explore: face > object
hypothesis(m.cnt.tar, "0 > cue1", alpha = 0.025)

## Hypothesis Tests for class b:
##          Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star
## 1 (0)-(cue1) > 0      0.04     0.02      0     0.07      31.09      0.97
## ---
## '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 did not reveal any credible differences associated with the population-level predictors, suggesting that saccades towards a target appearing at the previous location of the face are not more likely than saccades towards a target appearing at the previous location of the object.

Plots

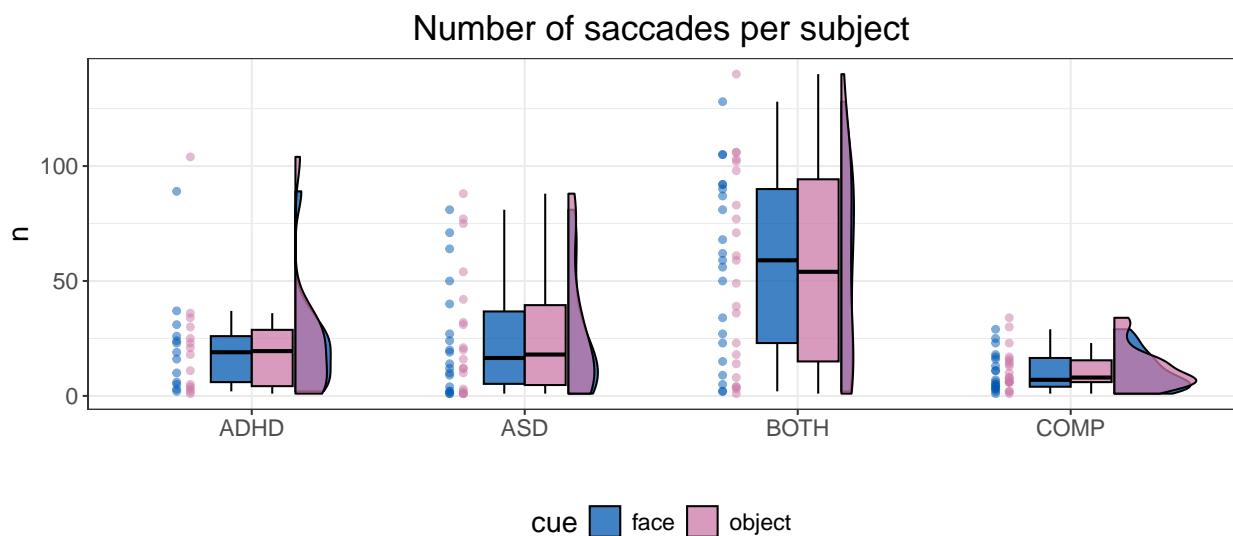
```
# rain cloud plot

df.cnt.tar %>%
  ggplot(aes(diagnosis, n.tar, fill = cue, colour = cue)) + #
  geom_rain(rain.side = 'r',
  boxplot.args = list(color = "black", outlier.shape = NA, show_guide = FALSE, alpha = 0.75),
  violin.args = list(color = "black", outlier.shape = NA, alpha = 0.75),
  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_dodge(nudge(x = -0.25, width = 0.1))) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  labs(title = "Number of saccades per subject", x = "", y = "n") +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal",
        text = element_text(size = 15))

```



Bayes factor analysis

To complement our hypothesis testing using `brms::hypothesis()`, we perform a Bayes Factor analysis with models excluding some of our population-level predictors.

```

# set the directory in which to save results
sense_dir = file.path(getwd(), "_brms_sens_cache")
log_dir = sense_dir
main.code = "cnt-tar"

# 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(2244)
  m.cnt.bf = brm(f.cnt,
    df.cnt.tar, prior = priors,
    iter = 40000, warmup = 10000,
    backend = "cmdstanr", threads = threading(8),
    file = "m_cnt-tar_bf",
    family = "poisson",
    save_pars = save_pars(all = TRUE)
  )
}

# loop through them
for (pr.desc in pr.descriptions) {
  tryCatch({
    # use the function
    bf_sens_2int(m.cnt.bf, "diagnosis", "cue", pr.desc,
      main.code, # prefix for all models and MLL
      file.path(log_dir, "log_FAB_bf.txt"), # log file
      sense_dir, # where to save the models and MLL
      reps = 5
    )
  },
  error = function(err) {
    message(sprintf("Error for %s: %s", pr.desc, err))
  }
)
}

# read in the results
df.cnt.bf = read_csv(file.path(sense_dir, sprintf("df_%s_bf.csv", main.code)),
  show_col_types = F)

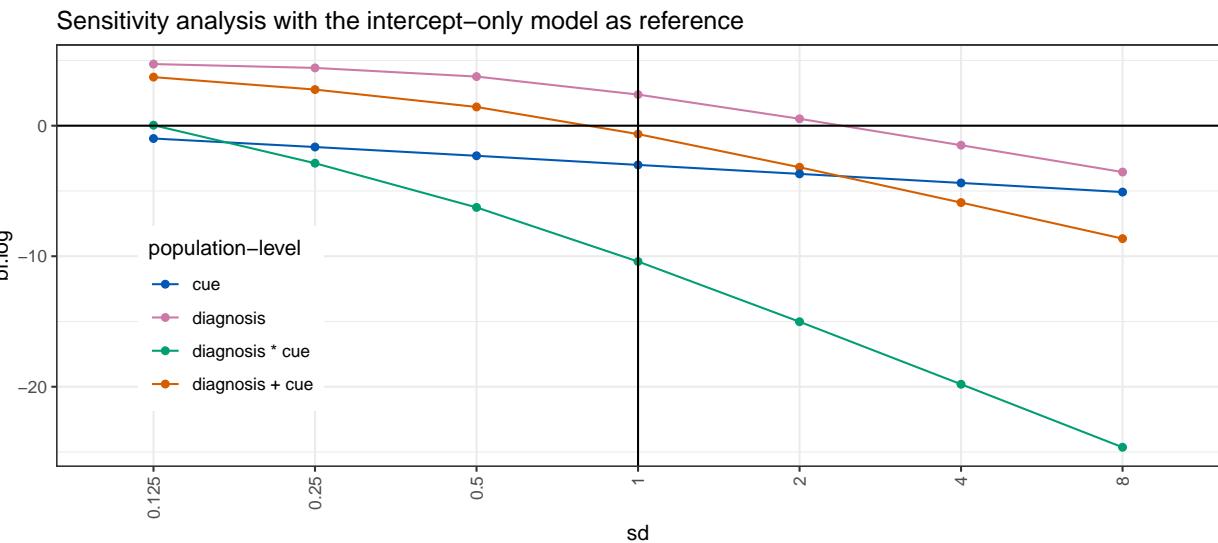
# check the sensitivity analysis result per model
df.cnt.bf %>%
  filter(`population-level` != "1") %>%
  mutate(
    sd = as.factor(case_when(
      priors == "chosen" ~ "1",
      substr(priors, 1, 3) == "sdx" ~ gsub("sdx", "", priors),
      T ~ priors
    )),
    order = case_when(
      priors == "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") +
#facet_wrap(. ~ `population-level`, scales = "free_y") +
scale_colour_manual(values = custom.col) +
theme_bw() +
theme(legend.position = c(0.15, 0.35),
      axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))

```



```

# print the BFs based on chosen priors
kable(df.cnt.bf %>% filter(priors == "chosen") %>% select(-priors) %>%
  filter(`population-level` != "1") %>% arrange(desc(bf.log)), digits = 3)

```

population-level	bf.log
diagnosis	2.387
diagnosis + cue	-0.643
cue	-3.004
diagnosis * cue	-10.405

This null effect was mirrored by the Bayes Factor analysis where the intercept-only model consistently outperforms the other models with the exception of very narrow priors (diagnostic groups: $\log(BF) = 2.387$; cue: $\log(BF) = -3.004$; diagnostic group and cue: $\log(BF) = -0.643$; full model: $\log(BF) = -10.405$).