S2: Analysis of FAB eye tracking data with brms

I. S. Plank

2025-02-05

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] "ggpubr version 0.6.0"
## [1] "ggrain version 0.0.4"
## [1] "bayesplot version 1.11.1"
## [1] "SBC version 0.3.0.9000"
## [1] "rstatix version 0.7.2"
## [1] "logspline version 2.1.22"
## [1] "BayesFactor version 0.9.12.4.7"
## [1] "bayestestR version 0.15.0"
```

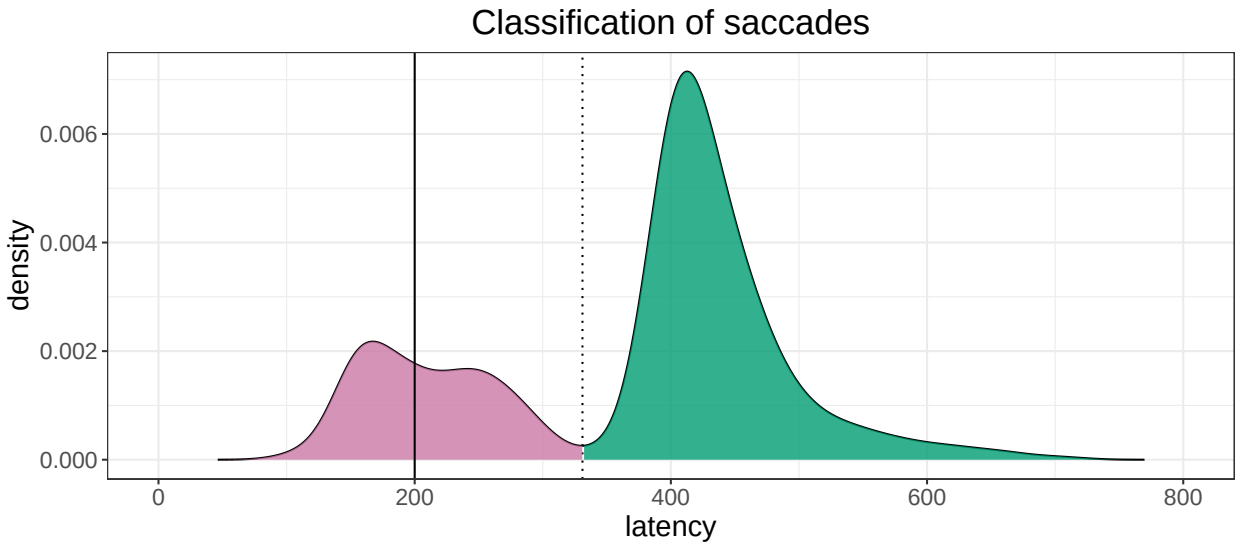
Preparation

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 use the global minimum 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)
```

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

# compute subject specific FAB effect size in cohen's d
df.fab.agg = df.fab %>%
  group_by(subID, diagnosis, stm, cue) %>%
  # summarise the median reaction time for each stimulus pair
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  ) %>%
  pivot_wider(names_from = cue, values_from = rt.cor) %>%
  group_by(subID, diagnosis) %>%
  # calculate the cohen's d
  summarise(
    fab = lsr::cohensD(x = object, y = face, method = "paired")
  )

# remove participants without any data
df.sac = df.sac %>% filter(!is.na(lat)) %>% ungroup() %>%
  # remove unbelievably short and extremely long saccades
  filter(lat <= quantile(lat, probs = 0.99) &
           lat > 100) %>%
  # recode so that it is more consistent
  mutate(
    direction = if_else(dir_face, "face", "object")
  )

# 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[2], colour = NA, alpha = .8) +
  geom_ribbon(data = subset(dd, x >= points$minima[idx]), aes(ymax = y), ymin = 0,
            fill = custom.col[3], colour = NA, alpha = .8) +
  geom_vline(xintercept = 200) +
  labs(title = "Classification of saccades", x = "latency", y = "density") +
  xlim(0, 800) +
  theme_bw() +
  theme(legend.position = "bottom",
        plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal",
        text = element_text(size = 15))
```




```
ggsave("Fig2_densLatency.pdf",
       units = "mm",
       width  = 170,
       height = 70,
       dpi    = 300)
```

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 minimum about 130ms 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.

```
# summarise overall saccade count based on direction: whole trial
df.cnt = df.sac %>%
  group_by(subID, direction) %>%
  summarise(
    n.sac = n()
  )

# add a zero if no saccades were produced
subID    = rep(as.character(unique(df.sac$subID)),
               each = length(unique(df.sac$direction)))
```

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

# code whether or not cue associated saccade occured and capture latencies
df.cue = merge(df.fab,
               df.sac %>% filter(lat <= points$minima[idx] & sac_trl == 1),
               all = T) %>%
  select(subID, diagnosis, trl, stm, cue, rt.cor, acc, direction, lat) %>%
  mutate(
    sac = if_else(is.na(direction),0,1)
  ) %>%
  mutate_if(is.character, as.factor)

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

# 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.cue$direction) = contr.sum(2)
contrasts(df.cue$direction)
```

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

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

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

```
contrasts(df.cnt$direction) = contr.sum(2)
contrasts(df.cnt$direction)
```

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

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.sac ~ diagnosis * direction + (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
```

Simulation-based calibration

```
# check if the SBC already exists
if (file.exists(file.path(cache_dir, sprintf("df_res_%s.rds", code)))) {
  # load in the results of the SBC
  df.results = readRDS(file.path(cache_dir, sprintf("df_res_%s.rds", code)))
  df.backend = readRDS(file.path(cache_dir, sprintf("df_div_%s.rds", code)))
  dat        = readRDS(file.path(cache_dir, sprintf("dat_%s.rds", code)))
} else {
  # 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")))
}
```

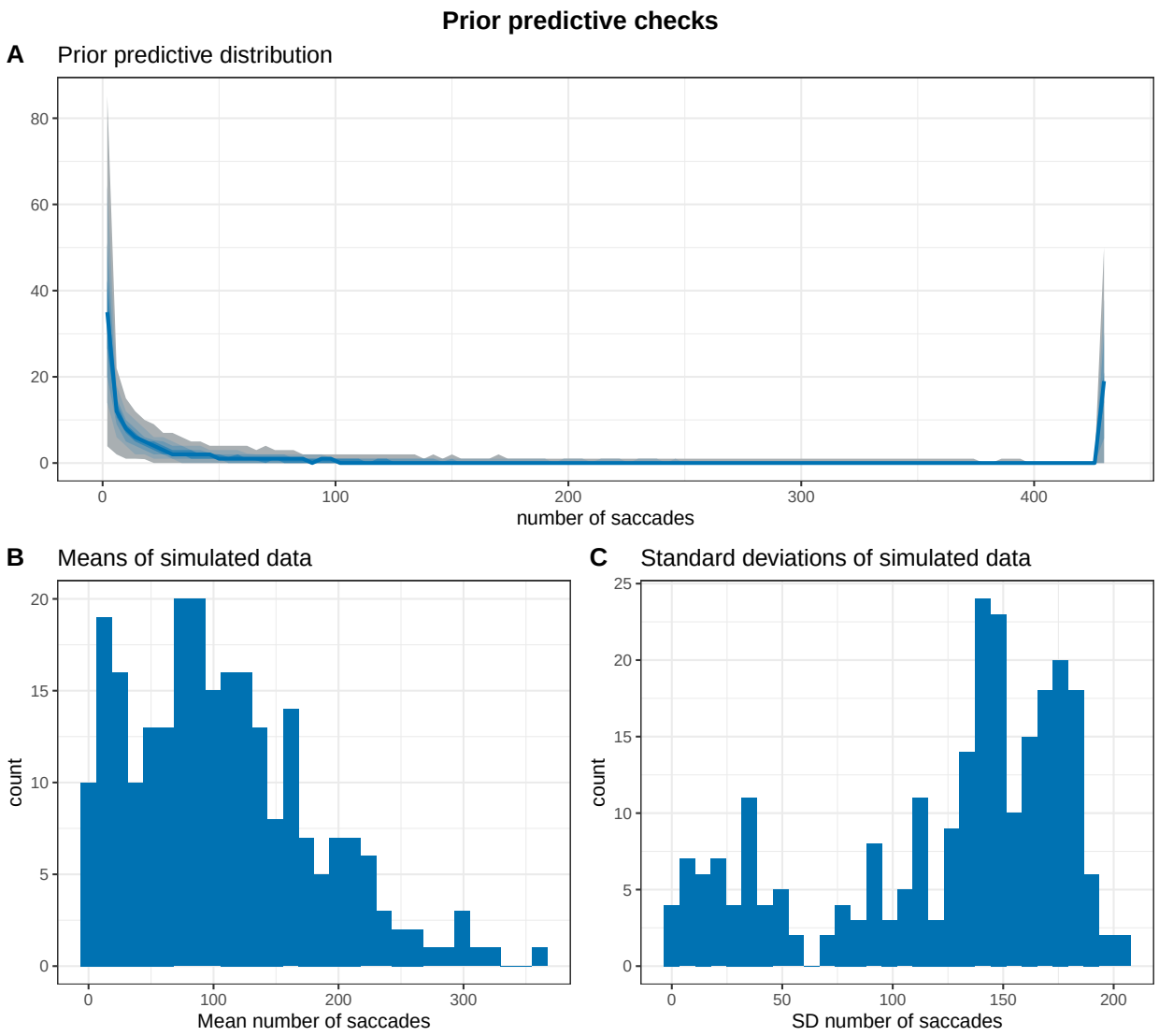
We start by investigating the rhats and the number of divergent samples. This shows that 6 of 250 simulations had at least one parameter that had an rhat of at least 1.05, and only 0 models had divergent samples. This suggests that this model performs well.

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(dvfakematH, 2, mean) # mean
tmpSD = apply(dvfakematH, 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))
```

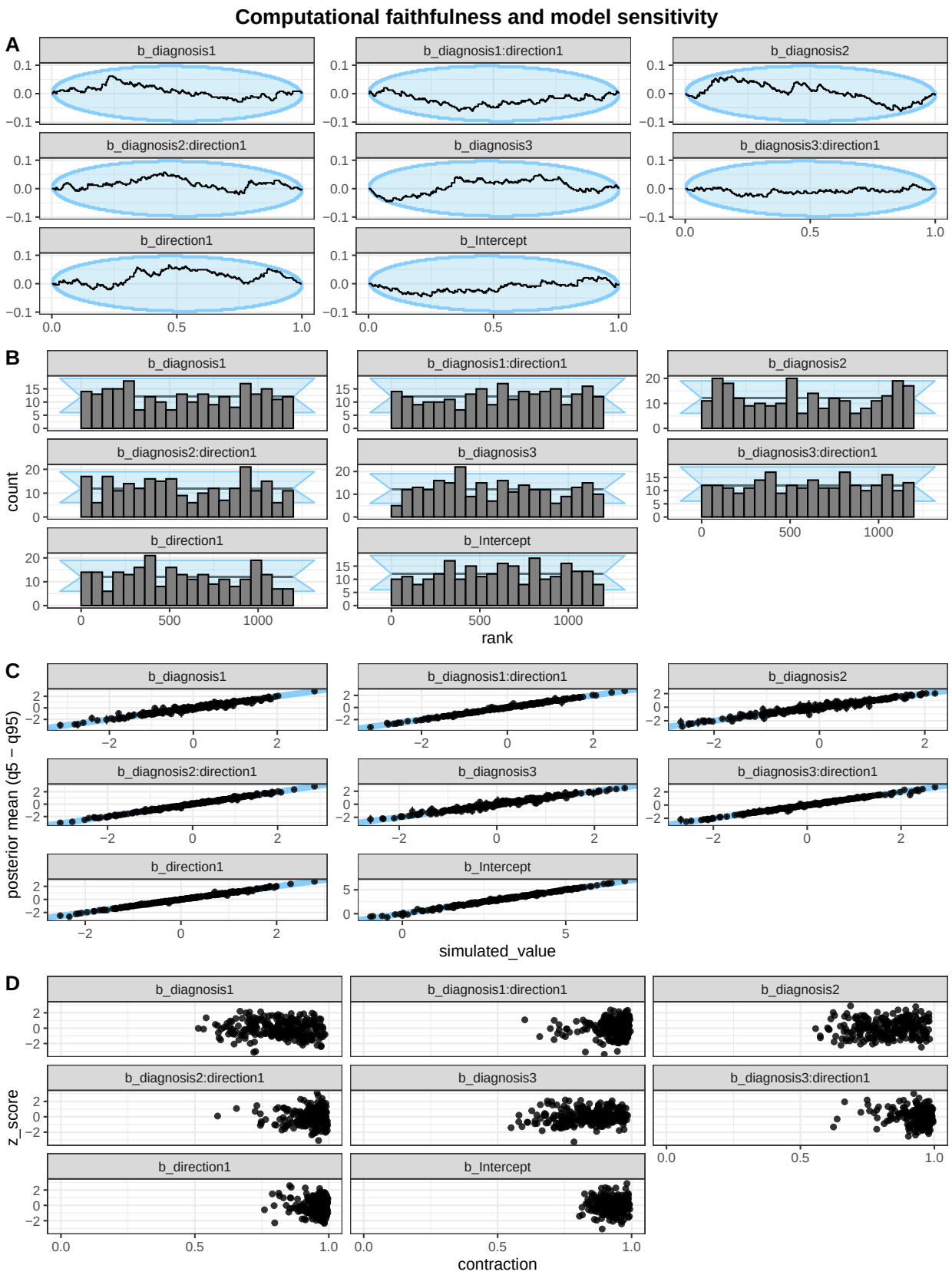


Since our priors were set very wide, we do get wide prior predictive distributions. We accept this and continue with our checks.

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
```

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



All of this looks good. 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:
```

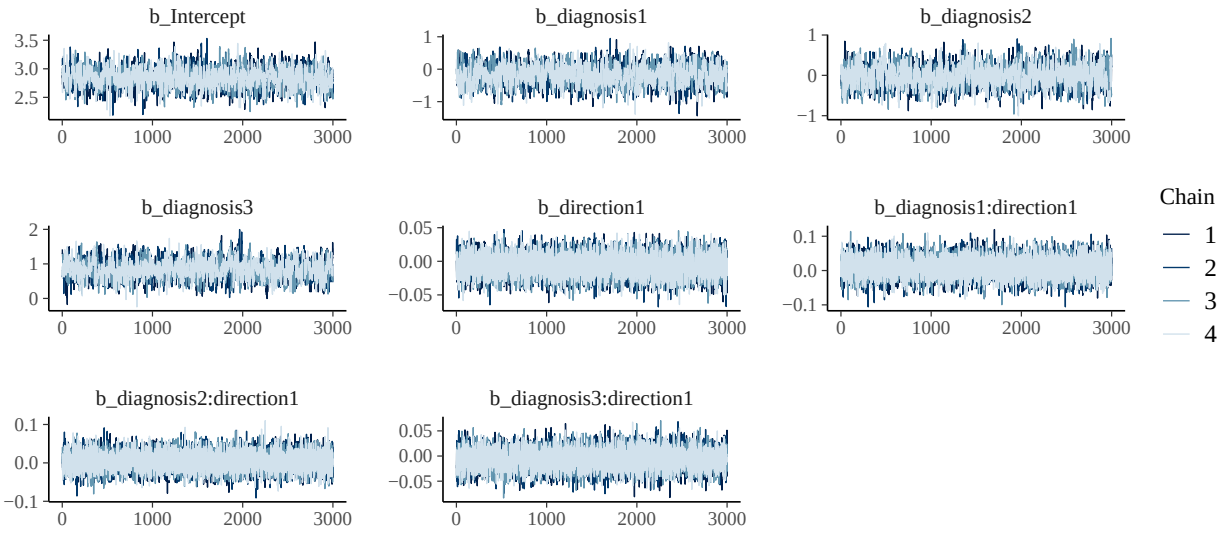
```
## E-BFMI indicated no pathological behavior.
```

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

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

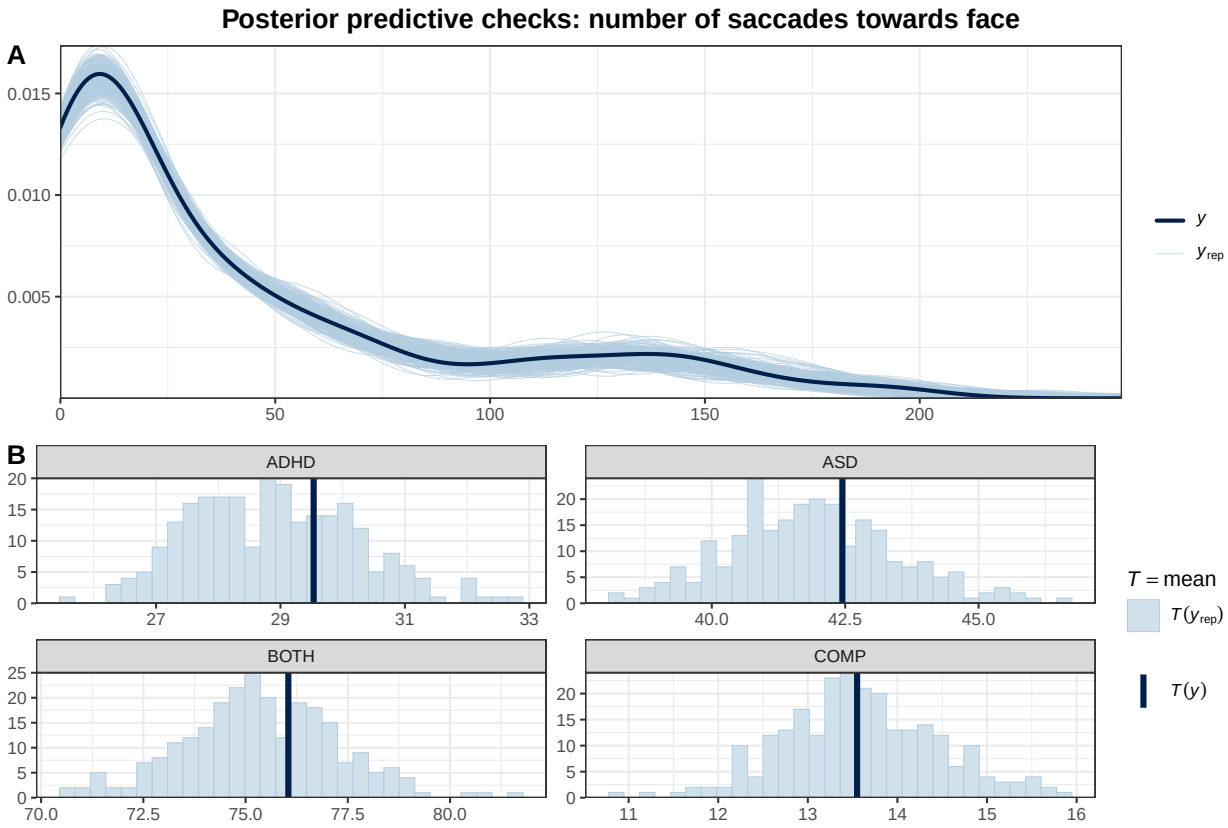
```
# check the trace plots
post.draws = as_draws_df(m.cnt)
mcmc_trace(post.draws, regex_pars = "^b_",
           facet_args = list(ncol = 3)) +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
```

```
## 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$n.sac, post.pred, df.cnt$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))
```



The predictions based on the model capture the data well. This further increases our trust in the model.

Inferences

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

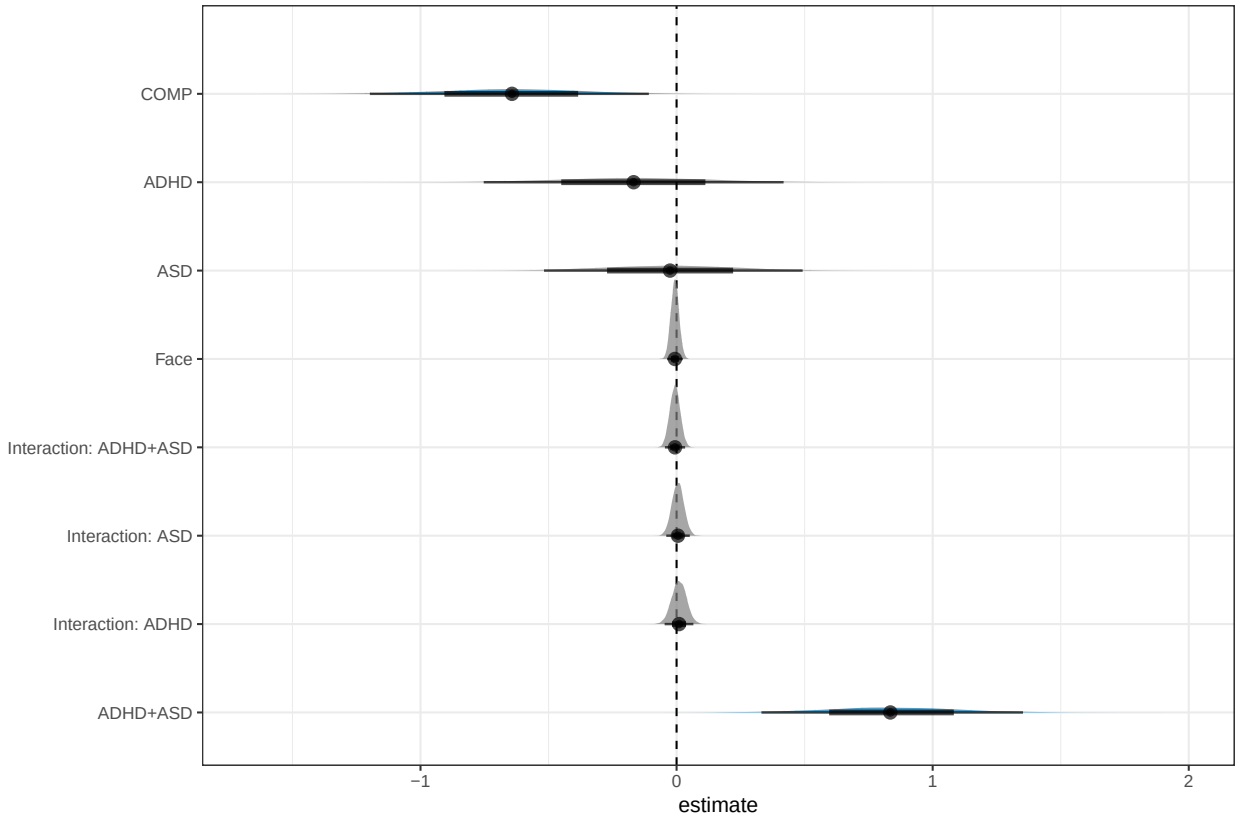
```
# print a summary
summary(m.cnt)
```

```
##  Family: poisson
##   Links: mu = log
## Formula: n.sac ~ diagnosis * direction + (1 | subID)
##    Data: df.cnt (Number of observations: 152)
##   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)     1.39      0.13     1.16     1.67 1.00     1831     3498
##
## Regression Coefficients:
##                      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                2.84      0.17     2.52     3.17 1.01     1329
## diagnosis1              -0.17      0.30    -0.75     0.42 1.00     1484
## diagnosis2              -0.02      0.26    -0.52     0.49 1.00     1339
## diagnosis3               0.84      0.26     0.33     1.35 1.01     1095
## direction1              -0.01      0.02    -0.04     0.02 1.00     7632
```

```
## diagnosis1:direction1     0.01      0.03    -0.05     0.07 1.00     8711
## diagnosis2:direction1     0.01      0.02    -0.04     0.05 1.00     9149
## diagnosis3:direction1    -0.01      0.02    -0.05     0.03 1.00     7972
##                       Tail_ESS
## Intercept                 2298
## diagnosis1                2349
## diagnosis2                2388
## diagnosis3                2245
## direction1                8645
## diagnosis1:direction1     8131
## diagnosis2:direction1     8479
## diagnosis3:direction1     8677
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
# get the estimates and compute groups
df.m.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,
    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") %>%
  mutate(
    coef = case_match(coef,
      "b_diagnosis1" ~ "ADHD",
      "b_diagnosis2" ~ "ASD",
      "b_diagnosis3" ~ "ADHD+ASD",
      "b_COMP"       ~ "COMP",
      "b_direction1" ~ "Face",
      "b_diagnosis1:direction1" ~ "Interaction: ADHD",
      "b_diagnosis2:direction1" ~ "Interaction: ASD",
      "b_diagnosis3:direction1" ~ "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 < 2*(direction1 - diagnosis1:direction1 -
                 diagnosis2:direction1 - diagnosis3:direction1)")
h2a
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*(direction... < 0     0.03      0.09    -0.11     0.17       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 general FAB
e1 = hypothesis(m.cnt, "0 < 2*direction1",
                alpha = 0.025)
e1
```

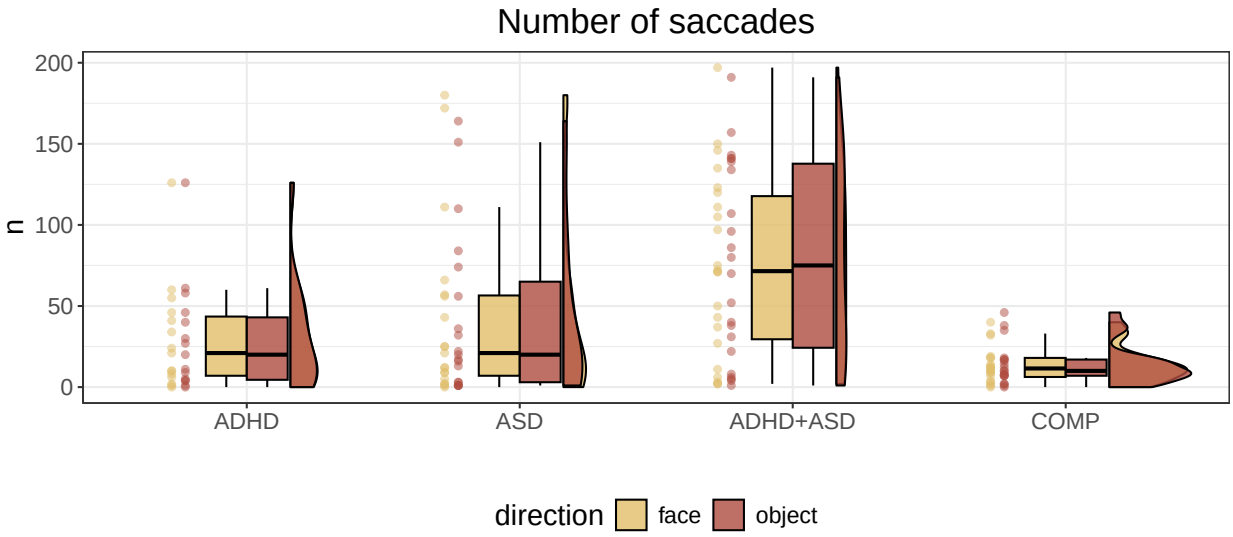
```
## Hypothesis Tests for class b:
##                Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*direction1) < 0     0.01      0.03    -0.05     0.07       0.51
##   Post.Prob Star
## 1      0.34
## ---
## 'CI': 95%-CI for one-sided and 97.5%-CI for two-sided hypotheses.
## '*': 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.
```

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 (*estimate* = 0.03 [-0.11, 0.17], *posterior probability* = 36.31%). Exploration of FAB effect regardless of the group similarly indicates no FAB in the form of a larger number of saccades produced towards the faces over the whole trial (*estimate* = 0.01 [-0.05, 0.07], *posterior probability* = 33.99%).

Plots

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

```
# rain cloud plot
df.cnt %>%
  mutate(
    diagnosis = recode(diagnosis, "BOTH" = "ADHD+ASD")
  ) %>%
  ggplot(aes(diagnosis, n.sac, fill = direction, colour = direction)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show_guide = FALSE, alpha = .8),
violin.args = list(color = "black", outlier.shape = NA, alpha = .8),
boxplot.args.pos = list(
  position = ggpp::position_dodgenudge(x = 0, width = 0.3), width = 0.3
),
point.args = list(show_guide = FALSE, alpha = .5),
violin.args.pos = list(
  width = 0.6, position = position_nudge(x = 0.16)),
point.args.pos = list(position = ggpp::position_dodgenudge(x = -0.25, width = 0.1))) +
  scale_fill_manual(values = custom.col2) +
  scale_color_manual(values = custom.col2) +
  labs(title = "Number of saccades", x = "", y = "n") +
  theme_bw() +
  theme(legend.position = "bottom",
        plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal",
        text = element_text(size = 15))
```




```
ggsave("Fig5_nrSac.pdf",
       units = "mm",
       width  = 170,
       height = 100,
       dpi    = 300)
```

S2.3 Latencies of target-associated saccades

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. 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). The SBC was run on three groups.

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")))
}
```

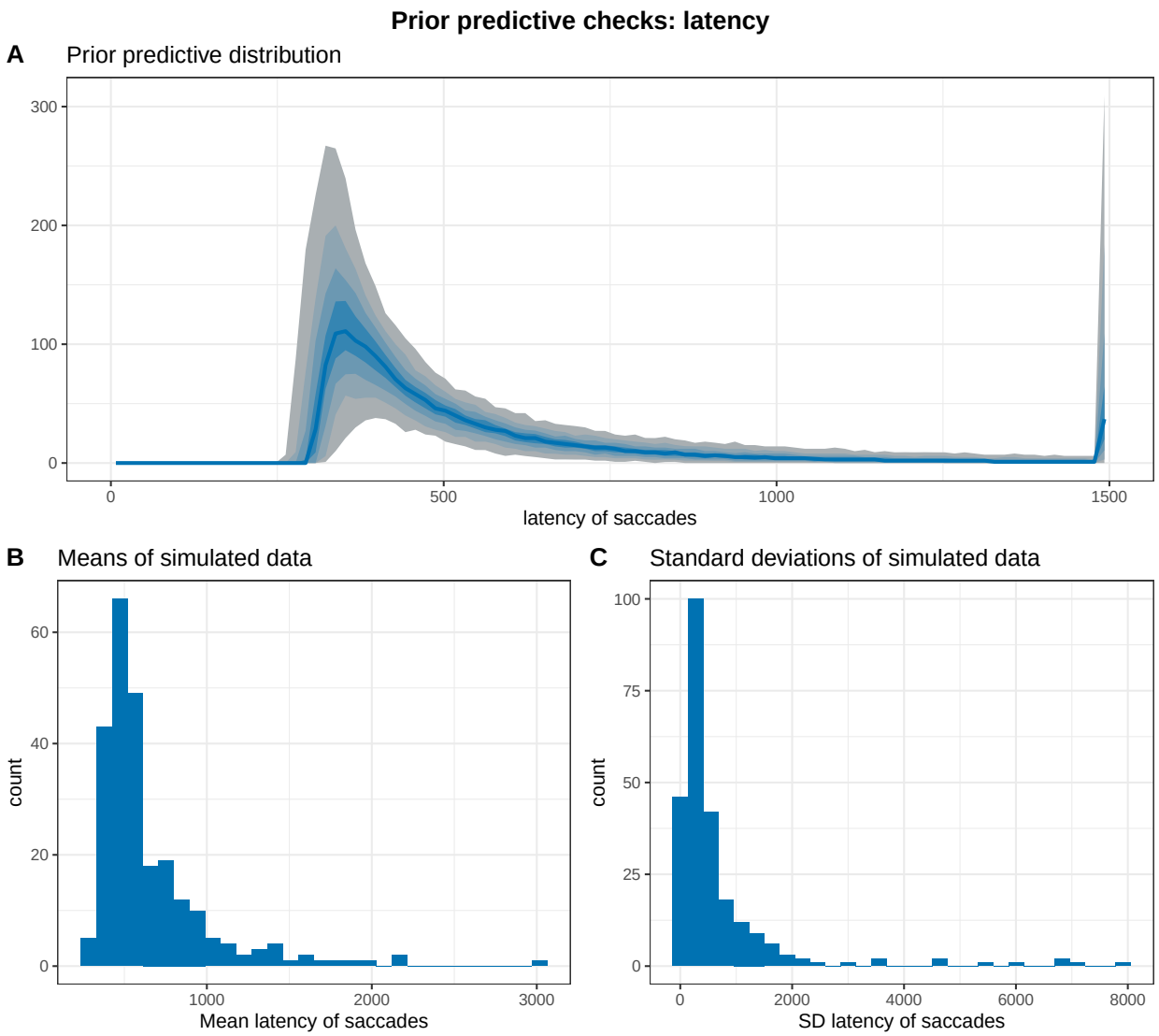
We start by investigating the rhats and the number of divergent samples. This shows that 0 of 250 simulations had at least one parameter that had an rhat of at least 1.05 and only 1 model had divergent samples (mean number of samples of the simulations with divergent samples: 1). This suggests that this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

```r
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))
```

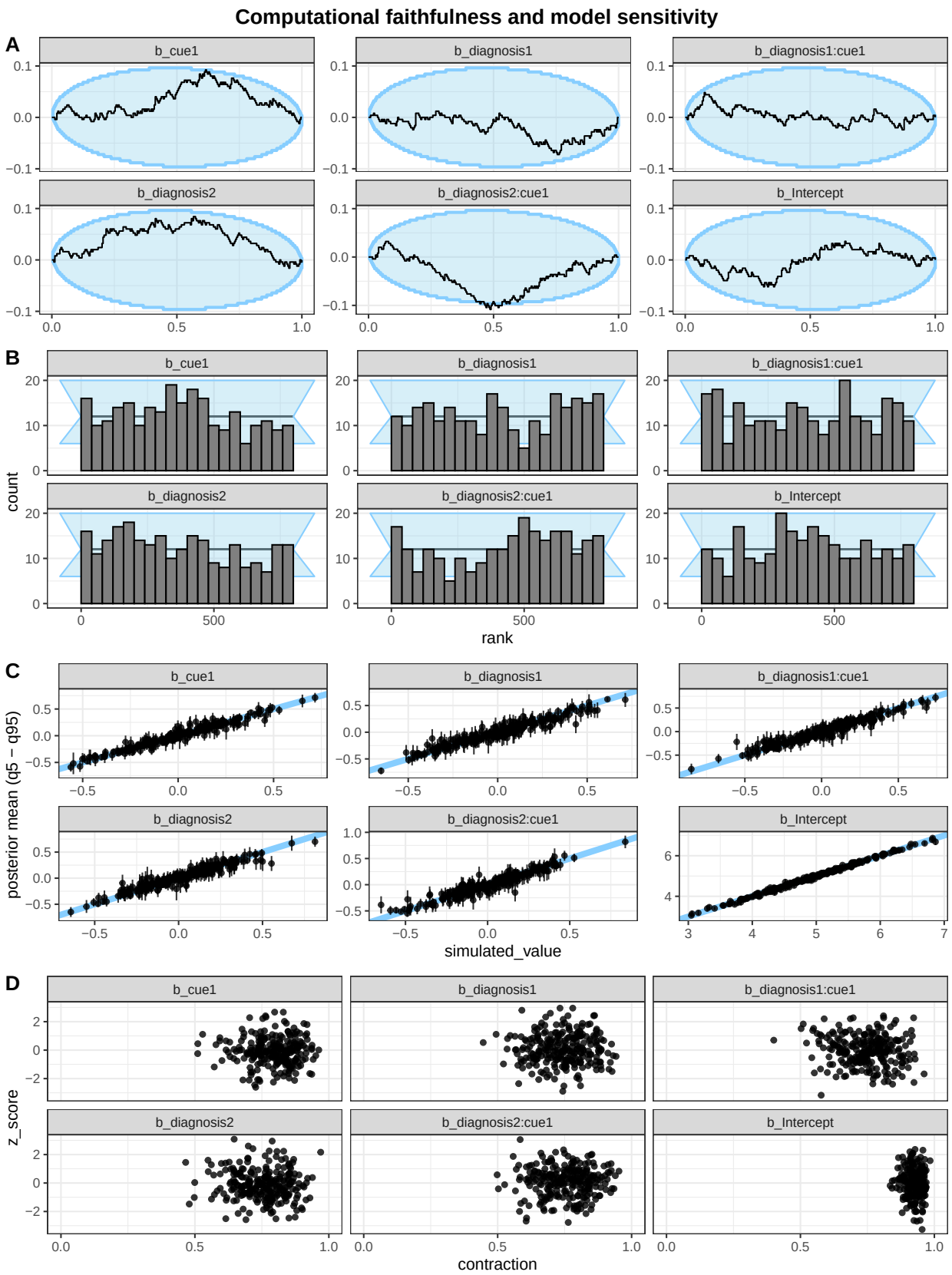


First, we assess whether the simulated values fit our expectations of the distribution of the data. Previous literature has found that saccade latencies are around 200ms with few saccades being produced faster than 100ms. If we add 200ms from the cue presentation, this means we expect most latencies 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)
```

```r
# 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$sim_id)) %>%
  ungroup() %>%
  mutate(
    max_rank = max(rank)
  )
p1 = plot_ecdf_diff(df.results.b) + theme_bw() + theme(legend.position = "none") +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p2 = plot_rank_hist(df.results.b, bins = 20) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p3 = plot_sim_estimated(df.results.b, alpha = .8) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p4 = plot_contraction(df.results.b,
                      prior_sd = setNames(
                        c(as.numeric(
                          gsub(".*, (.+)\\).*", "\\1",
                               priors[priors$class == "Intercept",]$prior)),
                          rep(
                            as.numeric(
                              gsub(".*, (.+)\\).*", "\\1",
                                   priors[priors$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))
```



All looks acceptable here.

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:
```

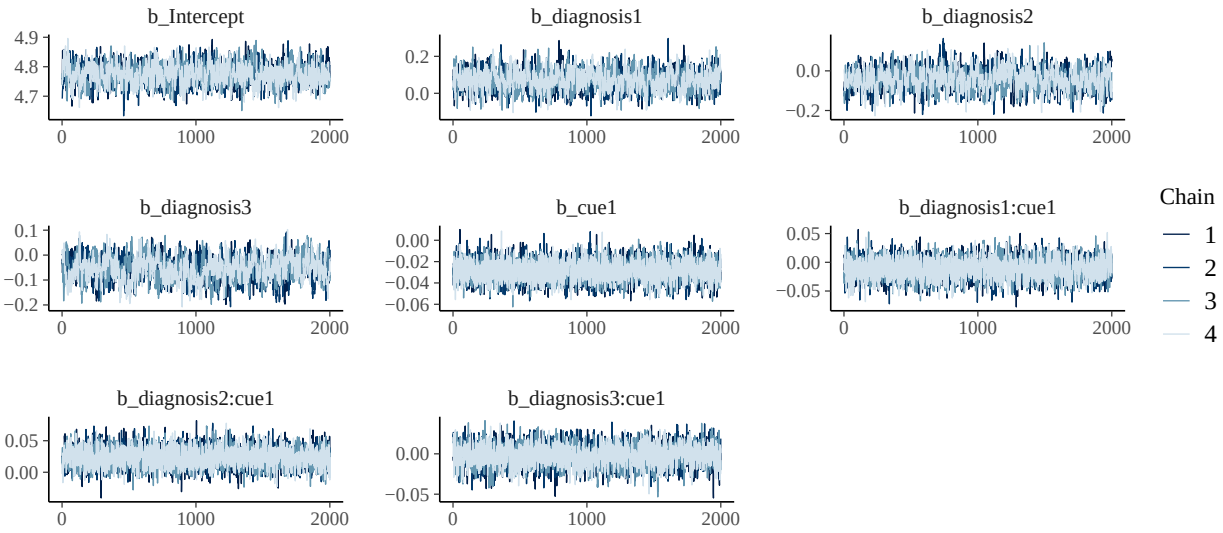
```
## E-BFMI indicated no pathological behavior.
```

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

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

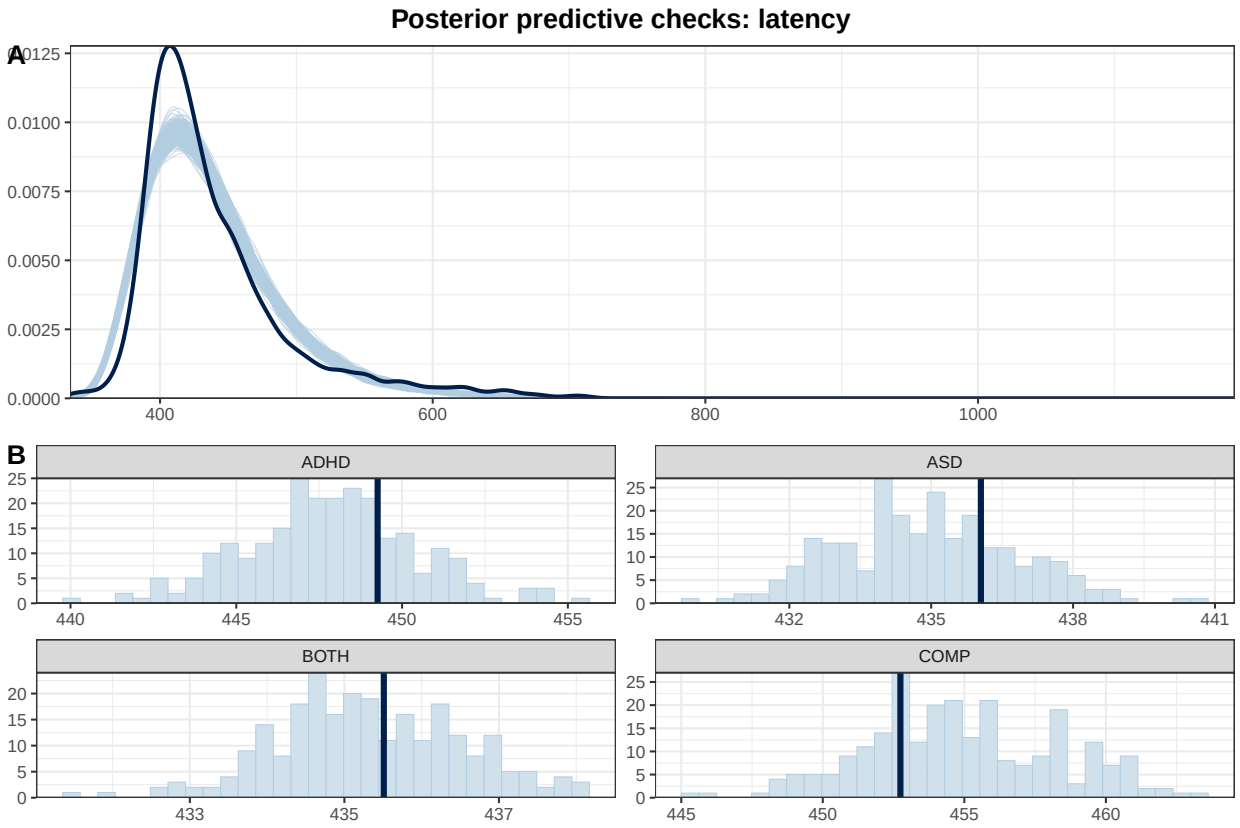
```
# check the trace plots
post.draws = as_draws_df(m.lat)
mcmc_trace(post.draws, regex_pars = "^b_",
           facet_args = list(ncol = 3)) +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
```

```
## 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))
```



The simulated data based on the model does not fit our data very well: it is wider and seems to underestimate latencies for COMP while overestimating for ADHD and 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
```

Again, we ran the SBC based on the three original, preregistered groups.

```
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")))
}
```

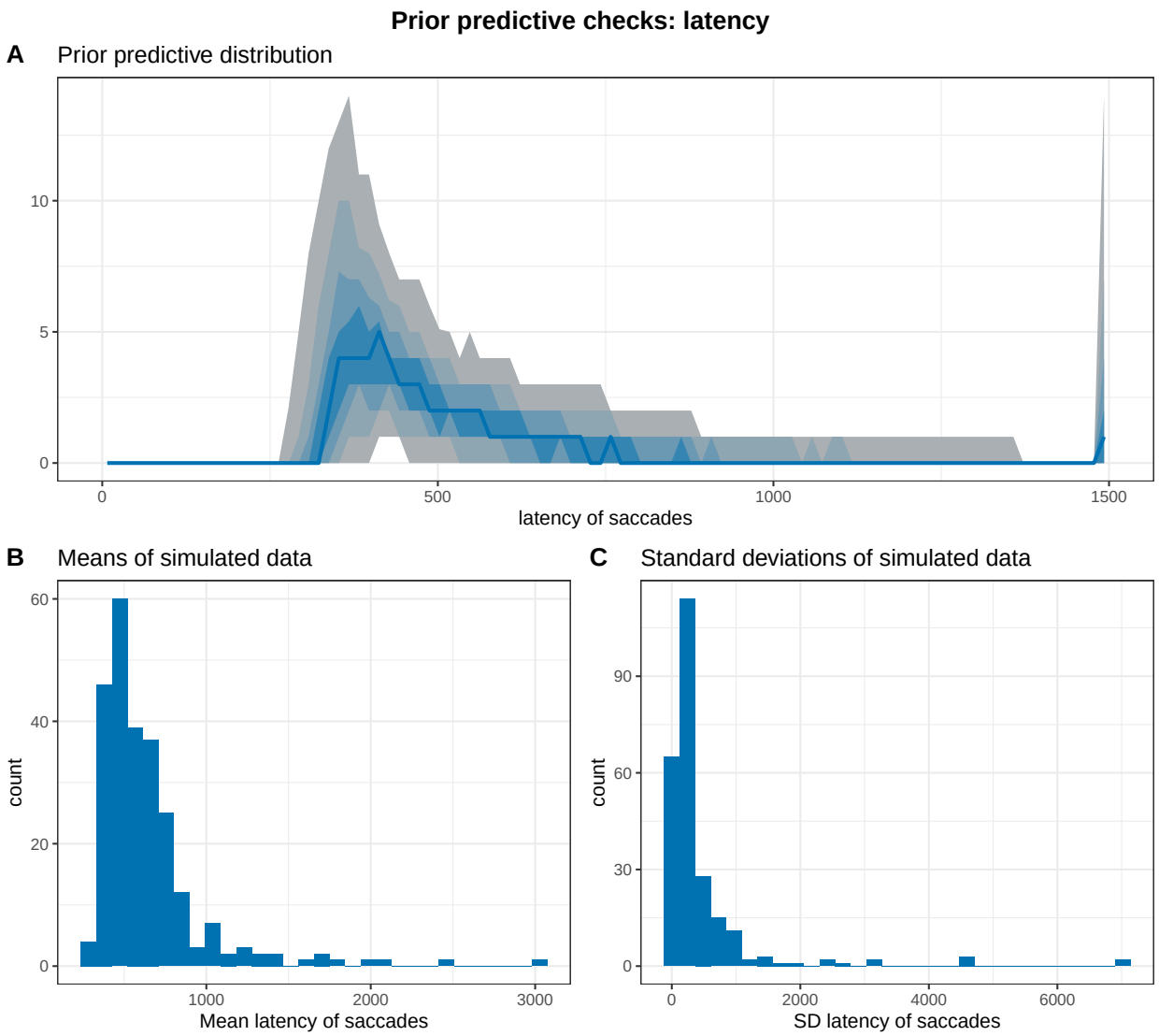
We start by investigating the rhats and the number of divergent samples. This shows that 1 of 250 simulations had at least one parameter that had an rhat of at least 1.05, but 105 models had divergent samples (mean number of samples of the simulations with divergent samples: 7.21). This is something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

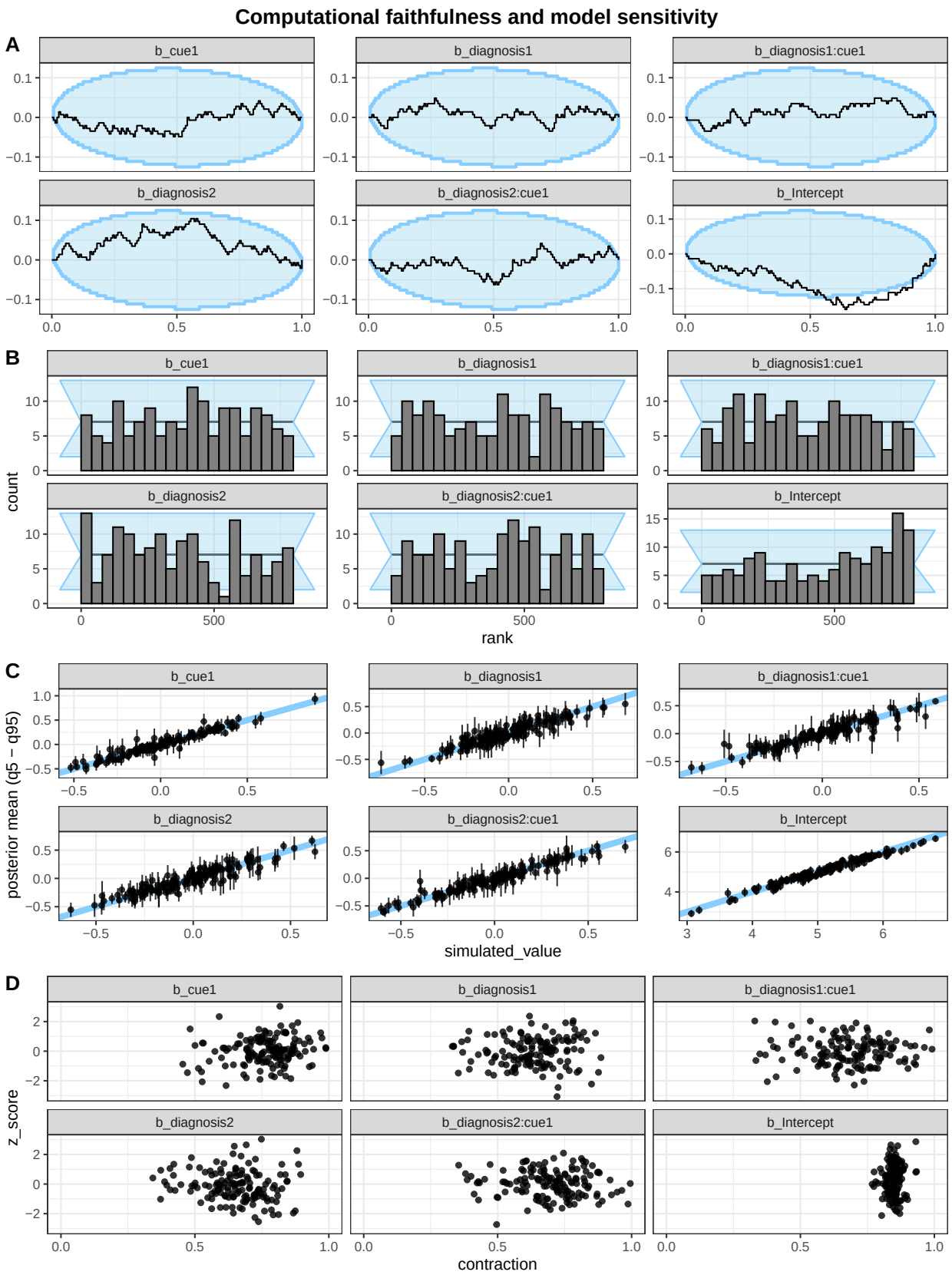


Again, our simulated datasets seem to capture well what we know about saccade latencies.

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$sim_id)) %>%
  ungroup() %>%
  mutate(
    max_rank = max(rank)
  )
p1 = plot_ecdf_diff(df.results.b) + theme_bw() + theme(legend.position = "none") +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p2 = plot_rank_hist(df.results.b, bins = 20) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p3 = plot_sim_estimated(df.results.b, alpha = .8) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p4 = plot_contraction(df.results.b,
                      prior_sd = setNames(
                        c(as.numeric(
                          gsub(".*, (.+)\\).*", "\\1",
                               priors[priors$class == "Intercept",]$prior)),
                          rep(
                            as.numeric(
                              gsub(".*, (.+)\\).*", "\\1",
                                   priors[priors$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))
```



The intercept looks slightly off here, the model could have a slight tendency to underestimate it.

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:
```

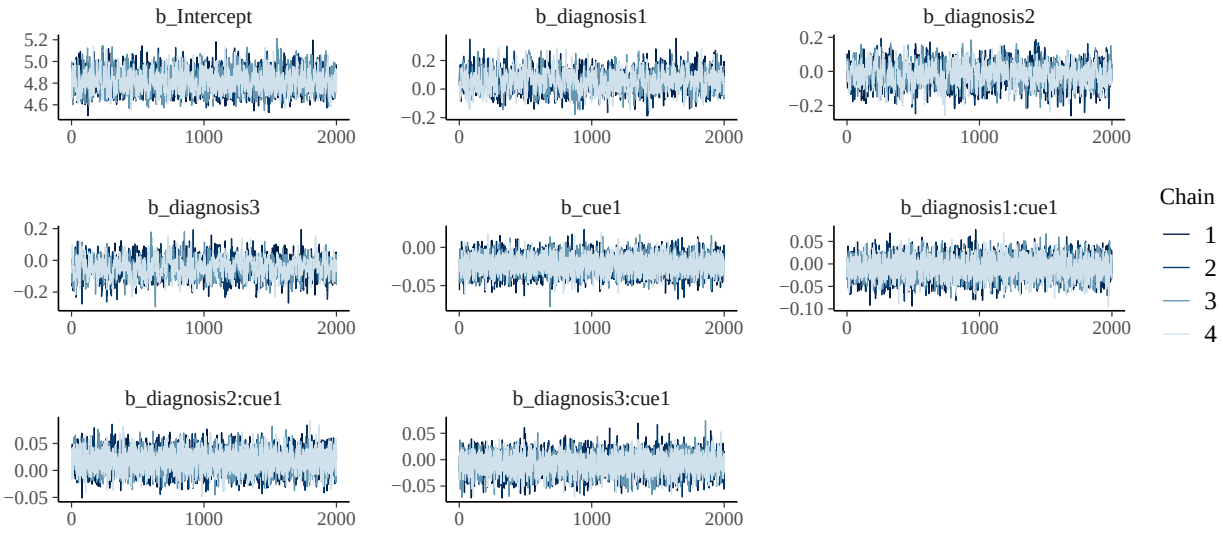
```
## E-BFMI indicated no pathological behavior.
```

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

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

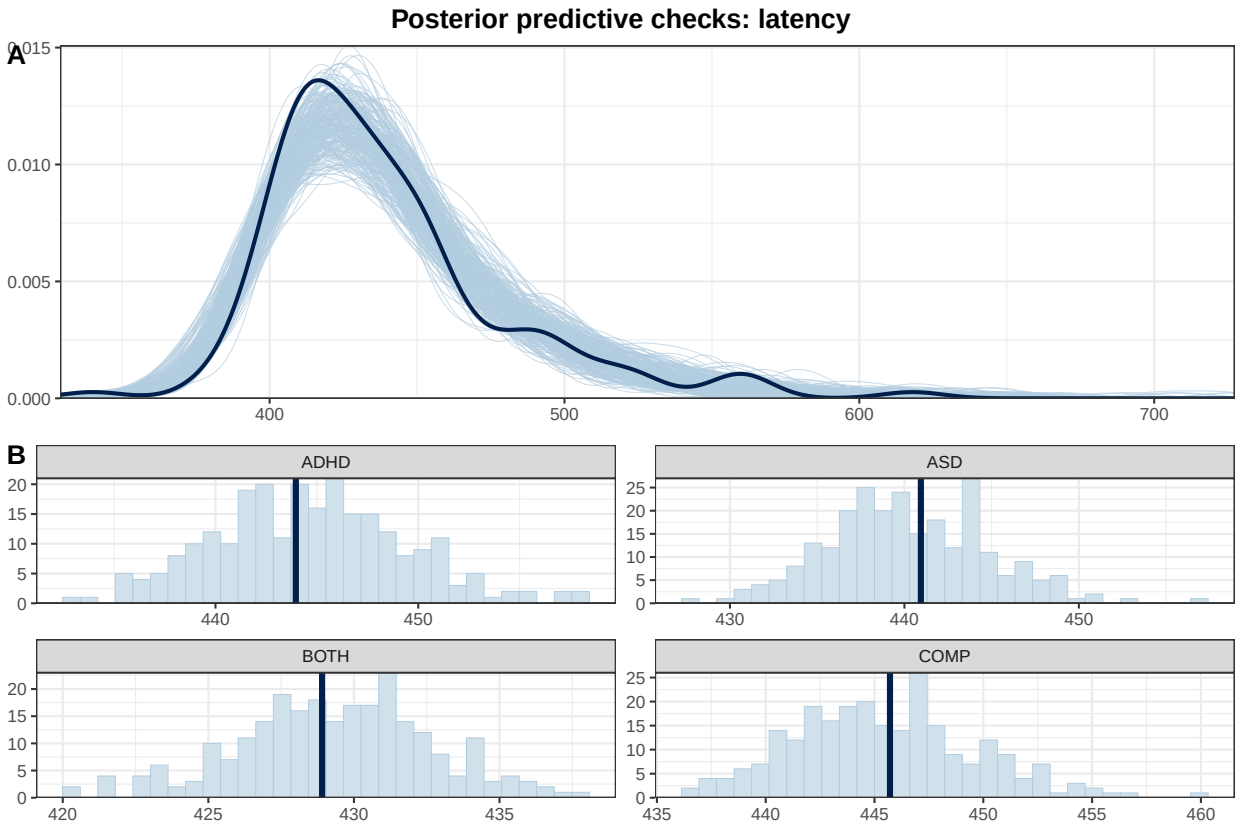
```
# check the trace plots
post.draws = as_draws_df(m.lat)
mcmc_trace(post.draws, regex_pars = "^b_",
           facet_args = list(ncol = 3)) +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
```

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



The final model does not exhibit any divergence issues or suboptimal rhats.

```
# 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))
```



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

Inferences

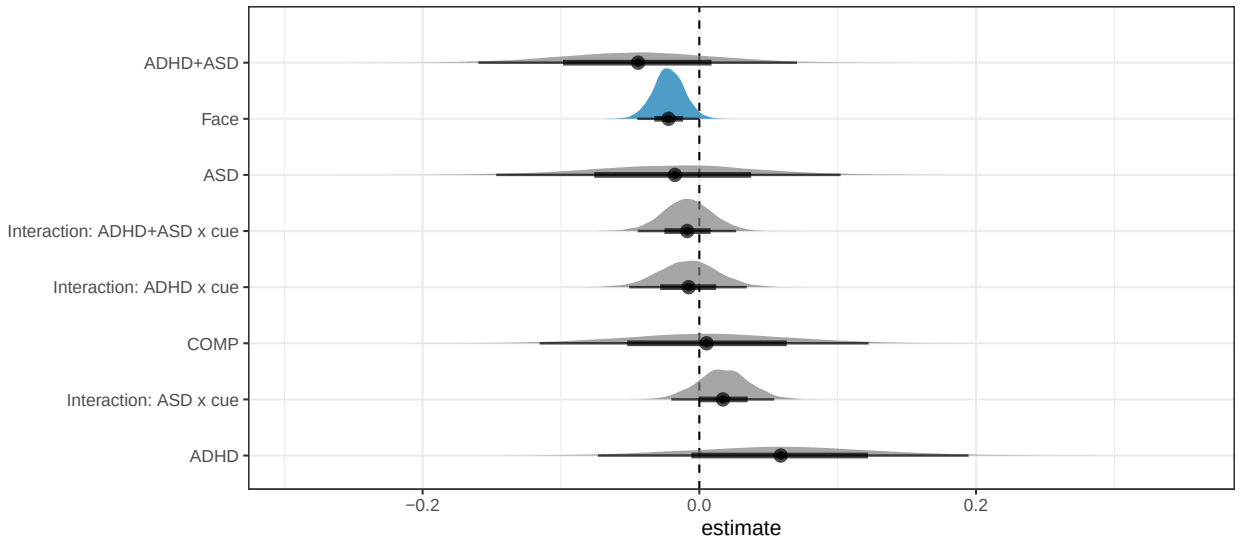
Now that we are convinced that we can trust our model, we have a look at the model and its estimates.

```
# 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 l-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).
```

```r
# 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, 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*diagnosis2:cue1")
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: overall faster towards face-cued targets
e = hypothesis(m.lat, "0 > 2*cue1", alpha = 0.025)
e
```

```
## Hypothesis Tests for class b:
##         Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob
## 1 (0)-(2*cue1) > 0     0.04      0.02        0     0.09       39.4      0.98
##   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.
```

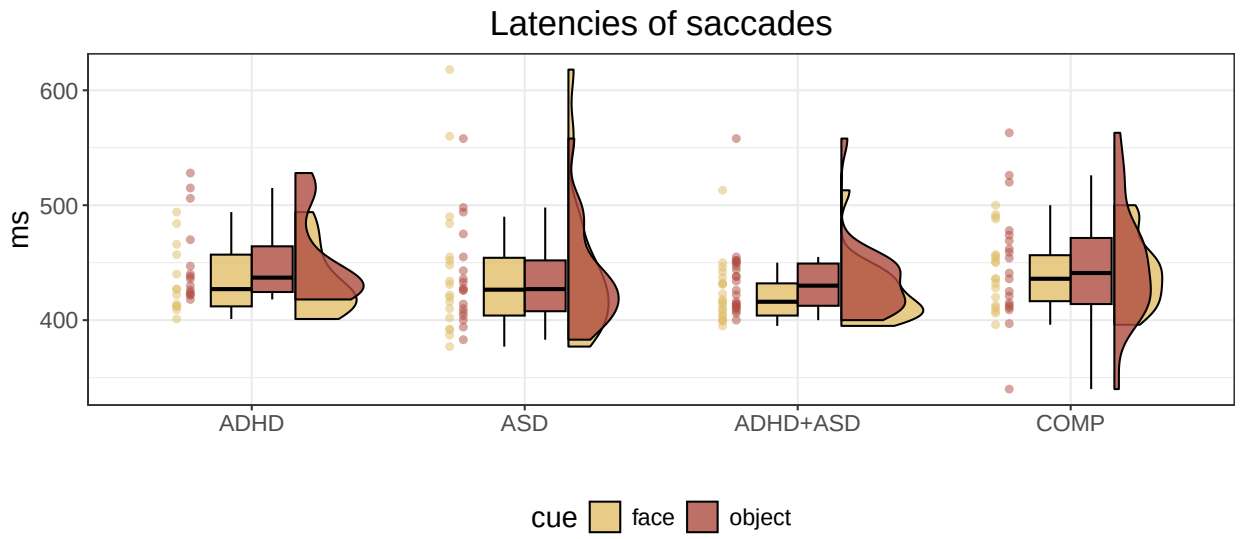
```
# extract predicted differences
df.new = df.lat %>%
  select(diagnosis, cue) %>%
  distinct() %>%
  mutate(
    condition = paste(diagnosis, cue, sep = "_")
```

```
  )
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(
    e  = rowMeans(select(., matches(".*_object")), na.rm = T) -
      rowMeans(select(., matches(".*_face")), na.rm = T)
  )
```

Our hypothesis that target-associated saccades towards the faces have a longer latency in ASD than COMP adults was not confirmed by the data (*estimate* = 0.01 [-0.17, 0.18], *posterior probability* = 47.24%). Our exploration revealed that latencies of saccades during a trial where the target appeared at the side of the face were faster regardless of the diagnostic group (*estimate* = 0.04 [0, 0.09], *posterior probability* = 97.52%). Specifically, the model predicted that if a saccade was produced towards a target appearing on the side of the face the latency was 5.556ms [0.008, 11.179] shorter than if a saccade was produced towards a target appearing at the side of the object cue.

Plots

```
# rain cloud plot for the
df.lat.agg %>%
  mutate(
    diagnosis = recode(diagnosis, "BOTH" = "ADHD+ASD")
  ) %>%
  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 = .8),
violin.args = list(color = "black", outlier.shape = NA, alpha = .8),
boxplot.args.pos = list(
  position = ggpp::position_dodgenudge(x = 0, width = 0.3), width = 0.3
),
point.args = list(show_guide = FALSE, alpha = .5),
violin.args.pos = list(
  width = 0.6, position = position_nudge(x = 0.16)),
point.args.pos = list(position = ggpp::position_dodgenudge(x = -0.25, width = 0.1))) +
  scale_fill_manual(values = custom.col2) +
  scale_color_manual(values = custom.col2) +
  labs(title = "Latencies of saccades", x = "", y = "ms") +
  theme_bw() +
  theme(legend.position = "bottom",
        plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal",
        text = element_text(size = 15))
```

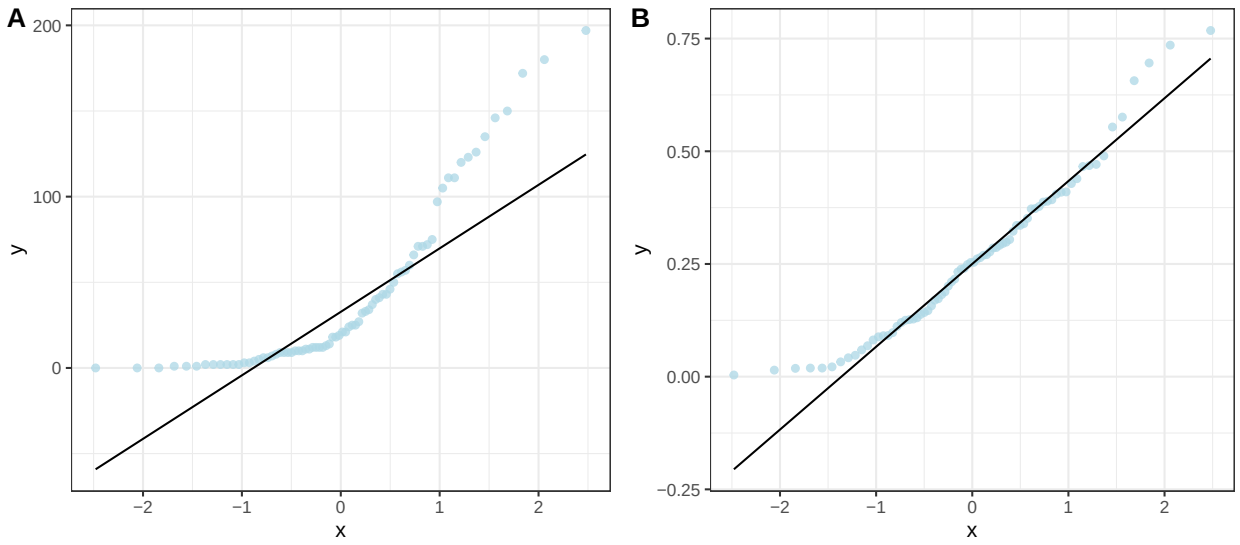



```
ggsave("Fig6_latSac.pdf",
       units = "mm",
       width  = 170,
       height = 100,
       dpi    = 300)
```

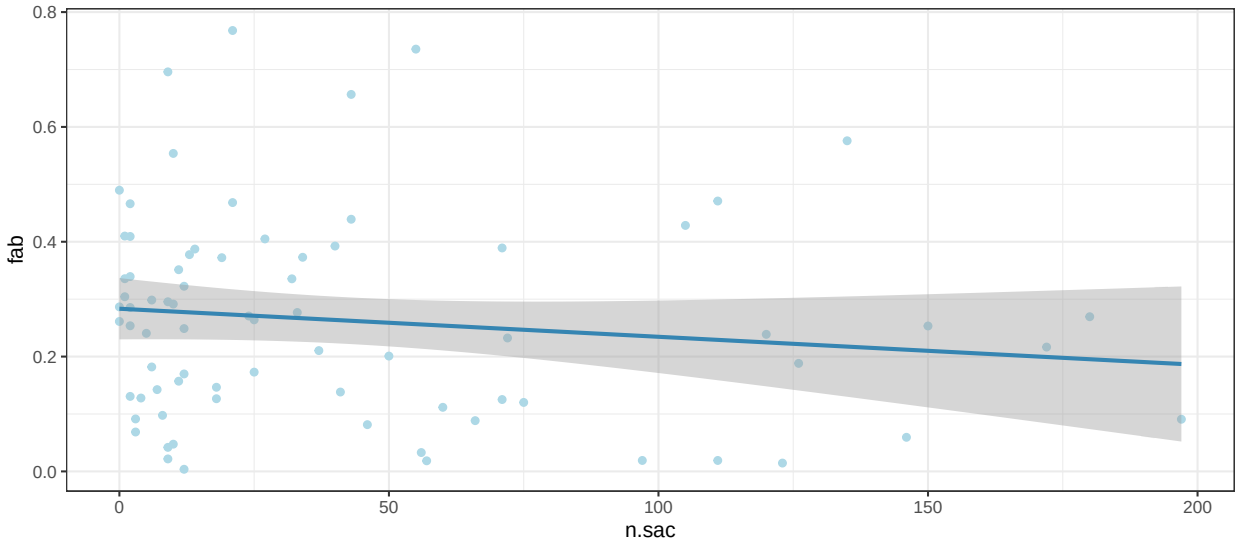
S2.4 Correlation with reaction times: number of saccades

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(direction == "face")

# check the distribution plot > not normally distributed
p1 = ggplot(df.diff, aes(sample = n.sac)) +
  stat_qq(alpha = 0.75, colour = "lightblue") +
  stat_qq_line() +
  theme_bw()
p2 = ggplot(df.diff, aes(sample = 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 {
  set.seed(1597)
  rhoSamples.cnt =
    spearmanGibbsSampler(xVals = df.diff$n.sac,
                         yVals = df.diff$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.sac, y = 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 FAB 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 anecdotal evidence against an association between the number of saccades and face attention bias ($\log(BF)$ = -0.741).

S2.5 Exploration: cue-associated saccades

Additionally to our hypotheses, we also explored any effects of diagnostic status, cue type and their interaction on whether a saccade was produced during the presentation of the cues to assess the findings of increased saccade frequency towards faces by Pereira et al. ([!ADD])in our data. Since these are again count data, we will use a Poisson and then compare the overall descriptives and effects to Pereira and colleague's results. Pereira found about 6% of trials contained cue-associated saccades which translates to an intercept of 3. Therefore, we can use the same priors and SBC as in our Poisson investigating numbers of saccades in general.

```
# aggregate to counts
df.cnt.cue = df.cue %>%
  group_by(subID, diagnosis) %>%
  summarise(
    face   = sum(direction == "face", na.rm = T),
    object = sum(direction == "object", na.rm = T)
  ) %>%
  pivot_longer(cols = c(face, object), names_to = "direction", values_to = "n.sac") %>%
  mutate_if(is.character, as.factor)

# aggregate for descriptives over both conditions
df.cnt.cue.agg = df.cnt.cue %>% group_by(subID, diagnosis) %>%
  summarise(n.sac = sum(n.sac))

# set the contrasts
contrasts(df.cnt.cue$direction) = contr.sum(2)
contrasts(df.cnt.cue$direction)
```

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

```
# set the same formula
f.cnt = brms::bf(n.sac ~ diagnosis * direction + (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
```

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(4682)
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:
```

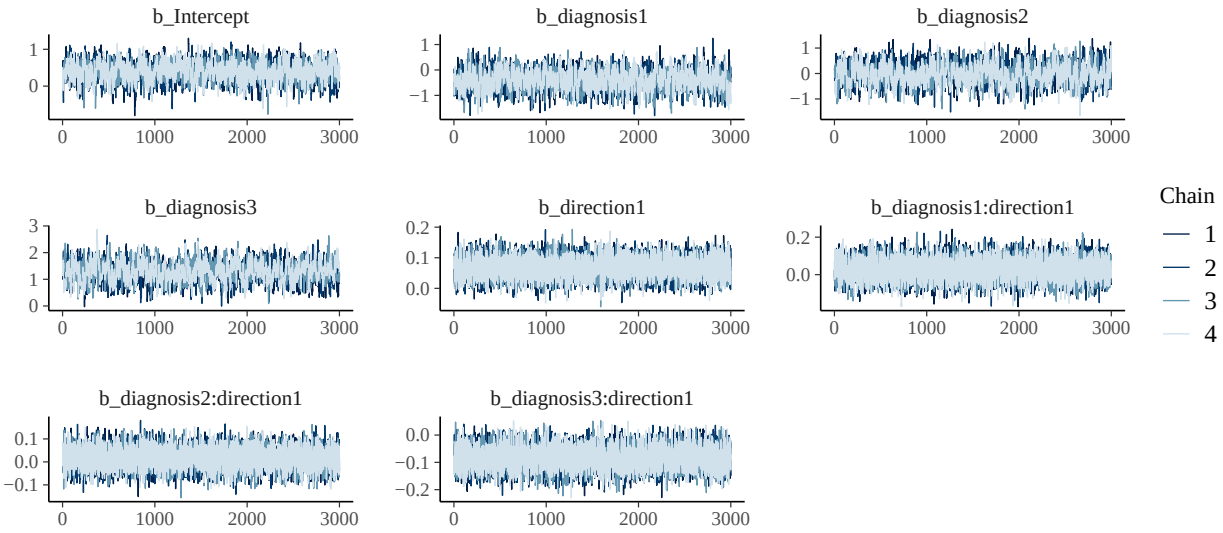
```
## E-BFMI indicated no pathological behavior.
```

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

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

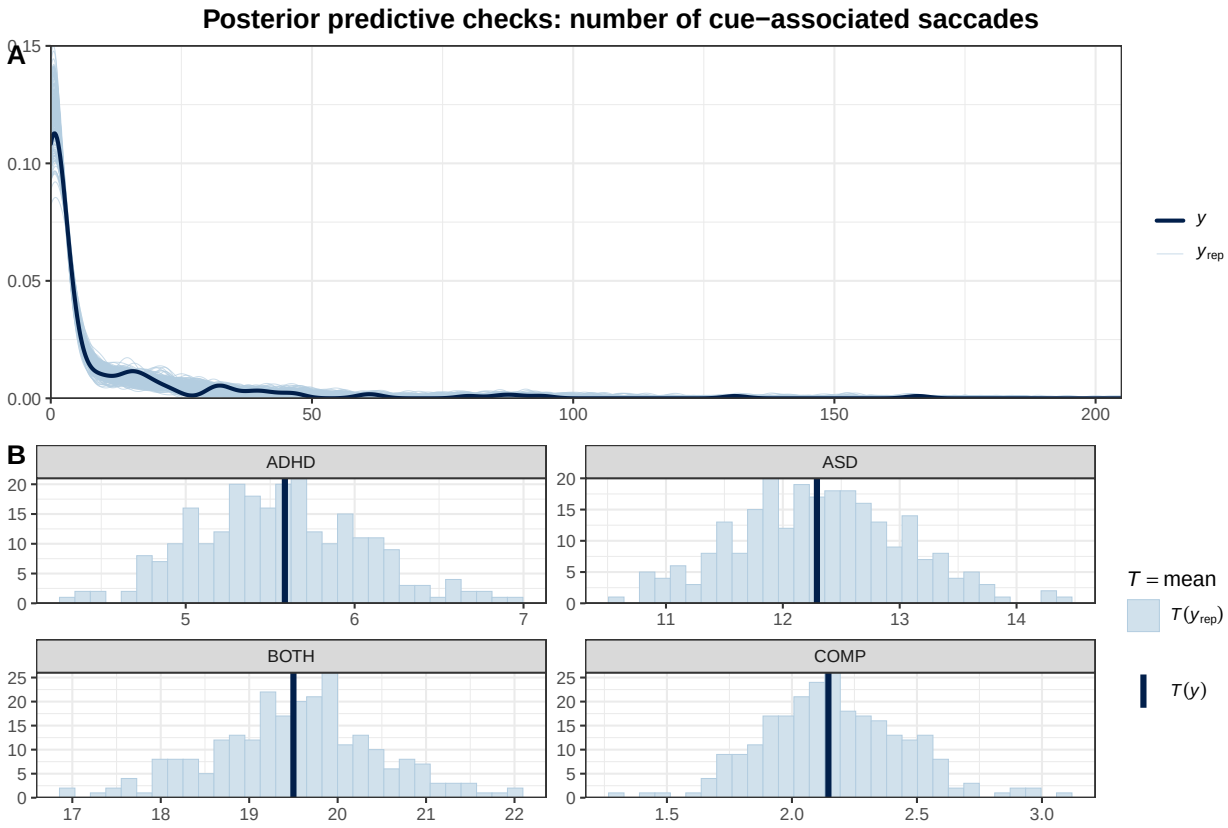
```
# check the trace plots
post.draws = as_draws_df(m.cnt)
mcmc_trace(post.draws, regex_pars = "^b_",
           facet_args = list(ncol = 3)) +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
```

```
## 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$n.sac, post.pred, df.cnt.cue$diagnosis) +
  theme_bw()

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



The predictions based on the model capture the data well. This further increases our trust in the model.

Inferences

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

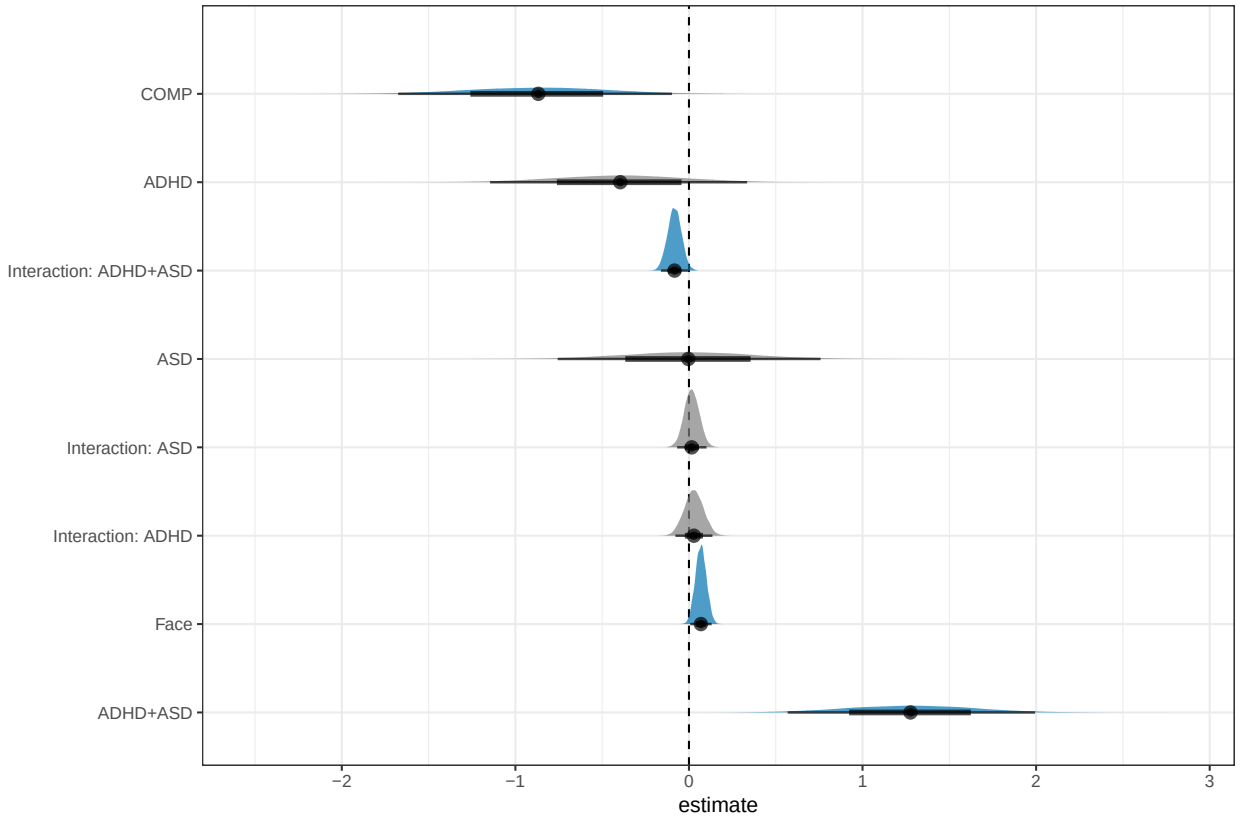
```
# print a summary
summary(m.cnt)
```

```
##  Family: poisson
##   Links: mu = log
## Formula: n.sac ~ diagnosis * direction + (1 | subID)
##    Data: df.cnt.cue (Number of observations: 188)
##   Draws: 4 chains, each with iter = 4500; warmup = 1500; thin = 1;
##          total post-warmup draws = 12000
##
## Multilevel Hyperparameters:
## ~subID (Number of levels: 94)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     2.12      0.21     1.74     2.57 1.00     1657     3655
##
## Regression Coefficients:
##                       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                 0.40      0.25    -0.10     0.87 1.00     1145
## diagnosis1               -0.40      0.38    -1.15     0.34 1.00     1375
## diagnosis2               -0.00      0.38    -0.76     0.76 1.01     1059
## diagnosis3                1.28      0.37     0.57     1.99 1.01      930
## direction1                0.07      0.03     0.01     0.13 1.00     7479
```

```
## diagnosis1:direction1     0.03      0.06    -0.08     0.14 1.00    11802
## diagnosis2:direction1     0.02      0.04    -0.07     0.10 1.00     9058
## diagnosis3:direction1    -0.08      0.04    -0.16    -0.01 1.00     8418
##                       Tail_ESS
## Intercept                 2593
## diagnosis1                2504
## diagnosis2                2057
## diagnosis3                2244
## direction1                9326
## diagnosis1:direction1     7984
## diagnosis2:direction1     8944
## diagnosis3:direction1     8315
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
# get the estimates and compute groups
df.m.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,
    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") %>%
  mutate(
    coef = case_match(coef,
      "b_diagnosis1" ~ "ADHD",
      "b_diagnosis2" ~ "ASD",
      "b_diagnosis3" ~ "ADHD+ASD",
      "b_COMP"       ~ "COMP",
      "b_direction1" ~ "Face",
      "b_diagnosis1:direction1" ~ "Interaction: ADHD",
      "b_diagnosis2:direction1" ~ "Interaction: ASD",
      "b_diagnosis3:direction1" ~ "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")
```




```
# face > object
e = hypothesis(m.cnt, "0 < 2*(direction1)",
               alpha = 0.025)
e
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*(direction1)) < 0    -0.14      0.06    -0.26    -0.01      67.57
##   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.
```

On average, participants produced cue-associated saccades on 4.55% +- 1.04 of the trials. However, the range was very wide, with some participants producing no cue-associated saccades and others producing them on 68.75% of the trials. Regardless of group, credibly more saccades were produced towards face than object cues (*estimate* = -0.14 [-0.26, -0.01], *posterior probability* = 98.54%).

S2.6 Exploration: Latencies of cue-associated saccades

We assume that the SBC for the hypothesis-guided latency analysis holds for this. We only need to slightly adjust the prior for the shift. This analysis only includes participants who performed cue-associated saccades.

```
# preprocess cue latencies
df.lat.cue = df.cue %>%
  filter(!is.na(direction)) %>%
  group_by(subID, direction, diagnosis) %>%
  summarise(lat = median(lat, na.rm = T)) %>%
  mutate_if(is.character, as.factor)

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

# 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(150, 50.00), class = ndt)  # this is the only prior that differs
)

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

# set the contrasts
contrasts(df.lat.cue$direction) = contr.sum(2)
contrasts(df.lat.cue$direction)
```

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

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

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

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

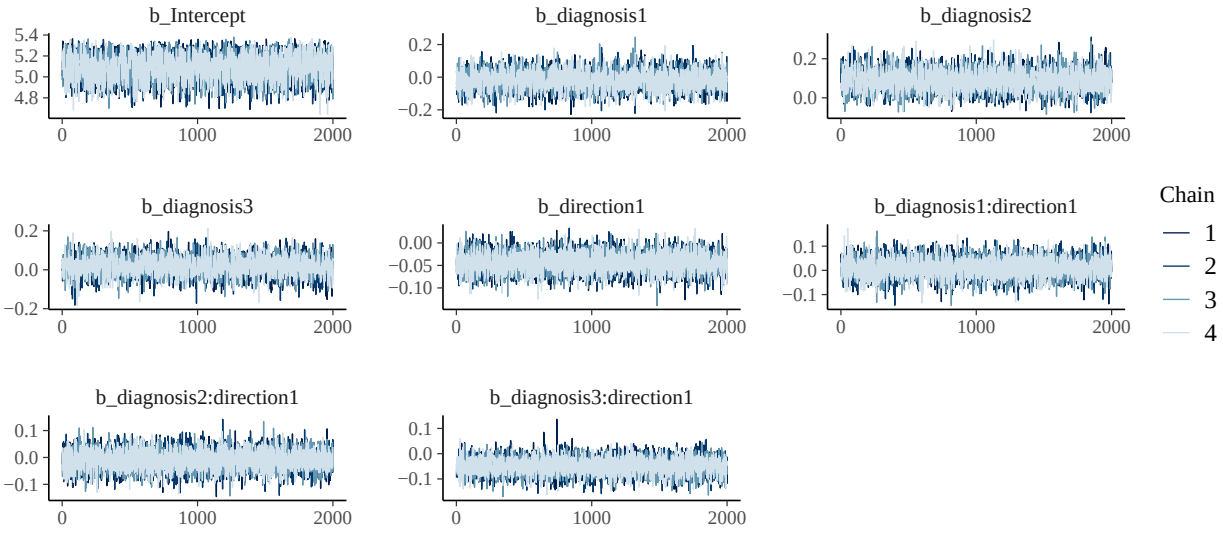
```
##
## Divergences:
## 0 of 8000 iterations ended with a divergence.
##
## Tree depth:
## 0 of 8000 iterations saturated the maximum tree depth of 10.
##
## Energy:
## E-BFMI indicated no pathological behavior.
```

```
# 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))
```

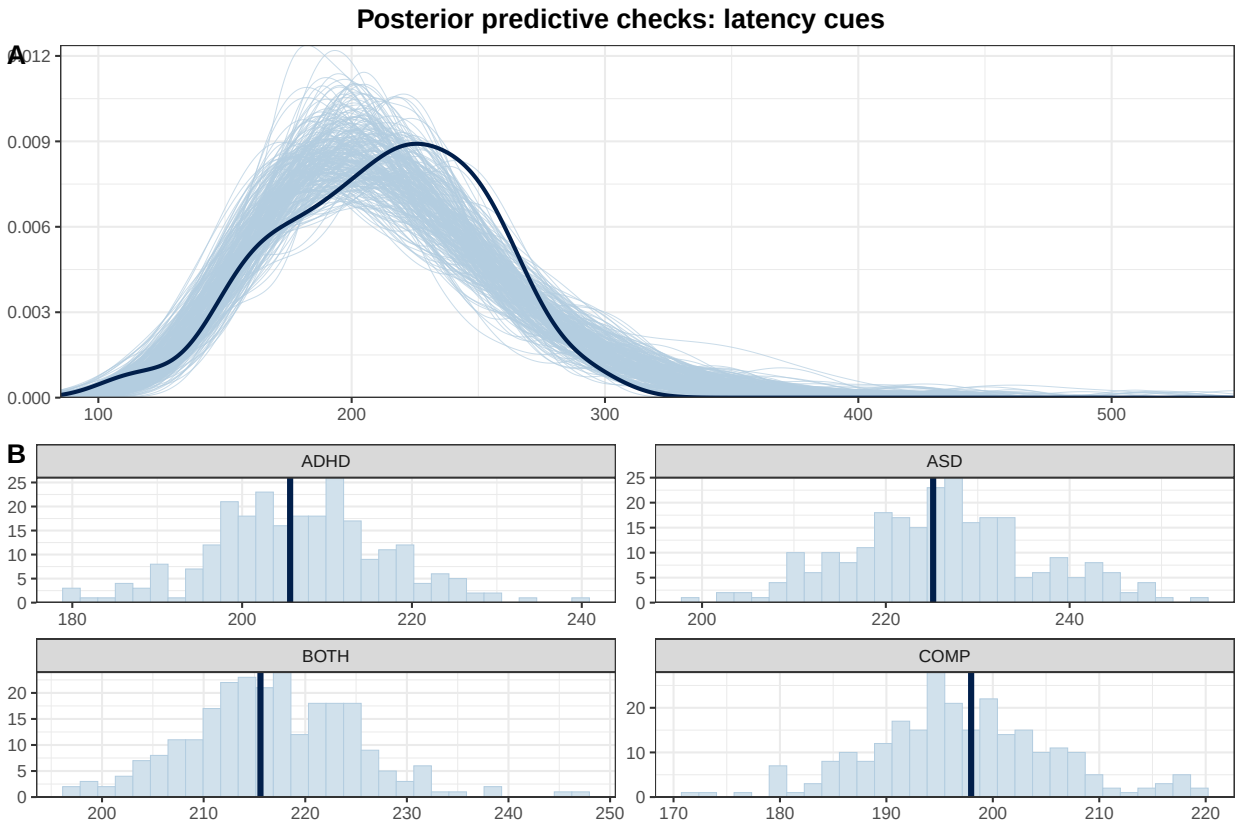
```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```



The final model does not exhibit any divergence issues or suboptimal rhats.

```
# 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.cue$lat, post.pred, df.lat.cue$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 cues",
                face = "bold", size = 14))
```



This looks good.

Inferences

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

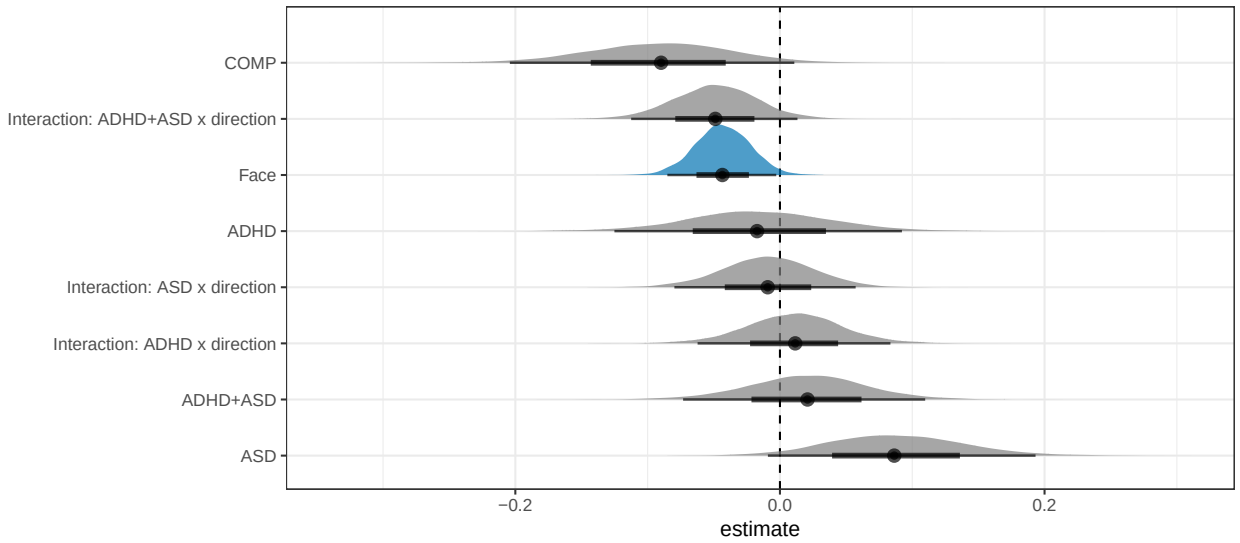
```
# print a summary
summary(m.lat)
```

```
##  Family: shifted_lognormal
##   Links: mu = identity; sigma = identity; ndt = identity
## Formula: lat ~ diagnosis * direction + (1 | subID)
##    Data: df.lat.cue (Number of observations: 115)
##   Draws: 4 chains, each with iter = 3000; warmup = 1000; thin = 1;
##          total post-warmup draws = 8000
##
## Multilevel Hyperparameters:
## ~subID (Number of levels: 64)
```

```
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     0.17      0.05     0.06     0.28 1.00     1249     1635
##
## Regression Coefficients:
##                       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                 5.09      0.12     4.85     5.31 1.00     3116
## diagnosis1               -0.02      0.05    -0.13     0.09 1.00     4137
## diagnosis2                0.09      0.05    -0.01     0.19 1.00     4364
## diagnosis3                0.02      0.05    -0.07     0.11 1.00     3717
## direction1               -0.04      0.02    -0.09    -0.00 1.00     7486
## diagnosis1:direction1     0.01      0.04    -0.06     0.08 1.00     7694
## diagnosis2:direction1    -0.01      0.03    -0.08     0.06 1.00     7773
## diagnosis3:direction1    -0.05      0.03    -0.11     0.01 1.00     8254
##                       Tail_ESS
## Intercept                 3595
## diagnosis1                4184
## diagnosis2                4666
## diagnosis3                5104
## direction1                5756
## diagnosis1:direction1     5542
## diagnosis2:direction1     5688
## diagnosis3:direction1     5502
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.20      0.04     0.14     0.29 1.00     1699     3667
## ndt      41.05     18.32     5.21    74.26 1.00     3142     3479
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
# 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_direction1" ~ "Face",
      "b_diagnosis1" ~ "ADHD",
      "b_diagnosis2" ~ "ASD",
      "b_diagnosis3" ~ "ADHD+ASD",
      "b_COMP" ~ "COMP",
      "b_diagnosis1:direction1" ~ "Interaction: ADHD x direction",
      "b_diagnosis2:direction1" ~ "Interaction: ASD x direction",
      "b_diagnosis3:direction1" ~ "Interaction: ADHD+ASD x direction"
    ),
    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")
```




```
# explore: faster towards faces
e = hypothesis(m.lat, "0 > 2*direction1", alpha = 0.025)
e
```

```
## Hypothesis Tests for class b:
##               Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*direction1) > 0     0.09      0.04     0.01     0.17      53.42
##   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.
```

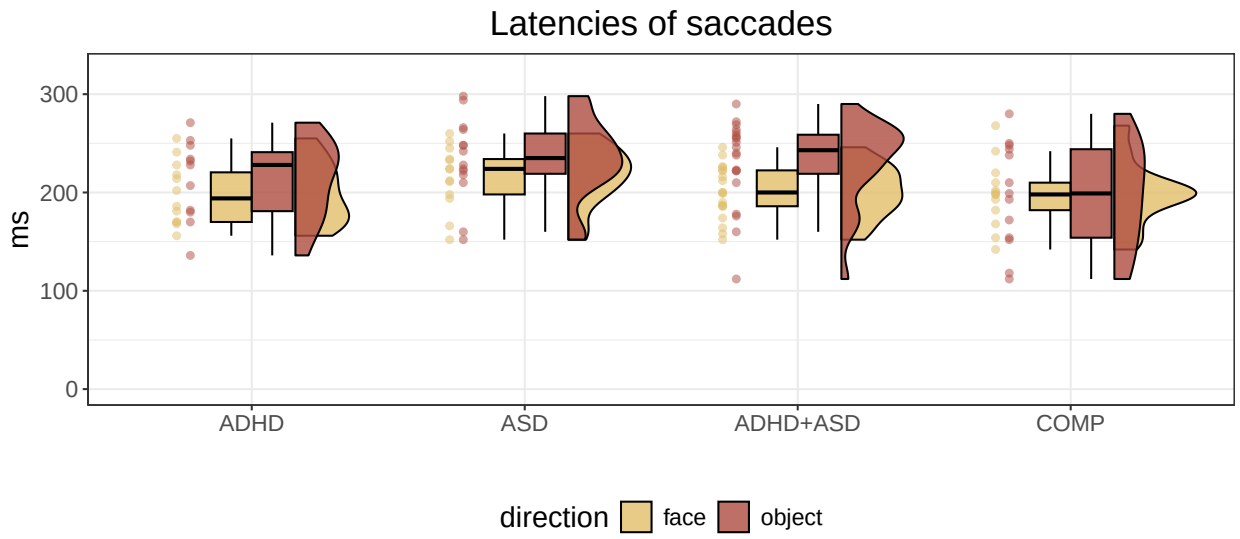
```
# extract predicted differences
df.new = df.lat.cue %>% ungroup() %>%
  select(diagnosis, direction) %>%
  distinct() %>%
  mutate(
    condition = paste(diagnosis, direction, sep = "_")
  )
df.ms = as.data.frame(
```

```
  fitted(m.lat, summary = F,
                newdata = df.new %>% select(diagnosis, direction),
                re_formula = NA))
colnames(df.ms) = df.new$condition

# calculate our difference columns
df.ms = df.ms %>%
  mutate(
    e  = rowMeans(select(., matches(".*_object")), na.rm = T) -
      rowMeans(select(., matches(".*_face")), na.rm = T)
  )
```

Cue-associated saccades were produces faster towards the face than the object cues (*estimate* = 0.09 [0.01, 0.17], *posterior probability* = 98.16%). Specifically, the model predicted that if a saccade was produced towards the face cue the latency was 14.915ms [1.412, 27.869] shorter than if a saccade was produced towards a target appearing at the side of the object cue.

Plots

```
# rain cloud plot for the
df.lat.cue %>%
  mutate(
    diagnosis = recode(diagnosis, "BOTH" = "ADHD+ASD")
  ) %>%
  ggplot(aes(diagnosis, lat, fill = direction, colour = direction)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show_guide = FALSE, alpha = .8),
violin.args = list(color = "black", outlier.shape = NA, alpha = .8),
boxplot.args.pos = list(
  position = ggpp::position_dodgenudge(x = 0, width = 0.3), width = 0.3
),
point.args = list(show_guide = FALSE, alpha = .5),
violin.args.pos = list(
  width = 0.6, position = position_nudge(x = 0.16)),
point.args.pos = list(position = ggpp::position_dodgenudge(x = -0.25, width = 0.1))) +
  scale_fill_manual(values = custom.col2) +
  scale_color_manual(values = custom.col2) +
  labs(title = "Latencies of saccades", x = "", y = "ms") +
  theme_bw() +
  ylim(0, 325) +
  theme(legend.position = "bottom",
        plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal",
        text = element_text(size = 15))
```


Latencies of saccades
300
200
100
0
ms
ADHD
ASD
ADHD+ASD
COMP
direction
face
object