S2: Analysis of FAB eye tracking data with brms

I. S. Plank

2025-07-02

# 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 4.5.0 (2025-04-11)"

## [1] "knitr version 1.50"  
## [1] "ggplot2 version 3.5.2"  
## [1] "brms version 2.22.0"  
## [1] "tidyverse version 2.0.0"  
## [1] "ggpubr version 0.6.0"  
## [1] "vtable version 1.4.8"  
## [1] "ggrain version 0.0.4"  
## [1] "bayesplot version 1.12.0"  
## [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] "effectsize version 1.0.1"  
## [1] "bayestestR version 0.16.0"

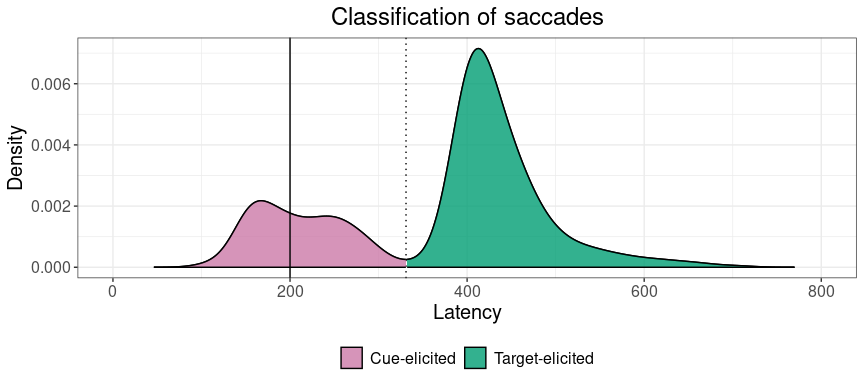
## Preparation

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)  
df.fab$diagnosis = factor(df.fab$diagnosis,   
 levels = c("ADHD", "ASD", "BOTH", "COMP"))  
  
# compute subject specific FAB   
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) %>%  
 # calculate the fab purely based on reaction times  
 mutate(  
 fab = object - face  
 ) %>% group\_by(subID, diagnosis) %>%  
 # calculate the mean FAB per person  
 summarise(  
 fab = mean(fab)  
 ) %>% ungroup()  
  
# 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)  
  
# print the latency  
points$minima[idx]

## [1] 331

# plot it all  
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, fill = "cue-elicited"), ymin = 0,   
 colour = "black", alpha = .8) +  
 geom\_ribbon(data = subset(dd, x >= points$minima[idx]),   
 aes(ymax = y, fill = "target-elicited"), ymin = 0,   
 colour = "black", alpha = .8) +  
scale\_fill\_manual(name = "test",  
values = c("cue-elicited" = custom.col[7], "target-elicited" = custom.col[3]),  
labels = c("Cue-elicited", "Target-elicited")) +   
 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),  
 legend.title=element\_blank())



ggsave("Fig2\_densLatency.svg",   
 units = "mm",   
 width = 170,  
 height = 100,  
 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

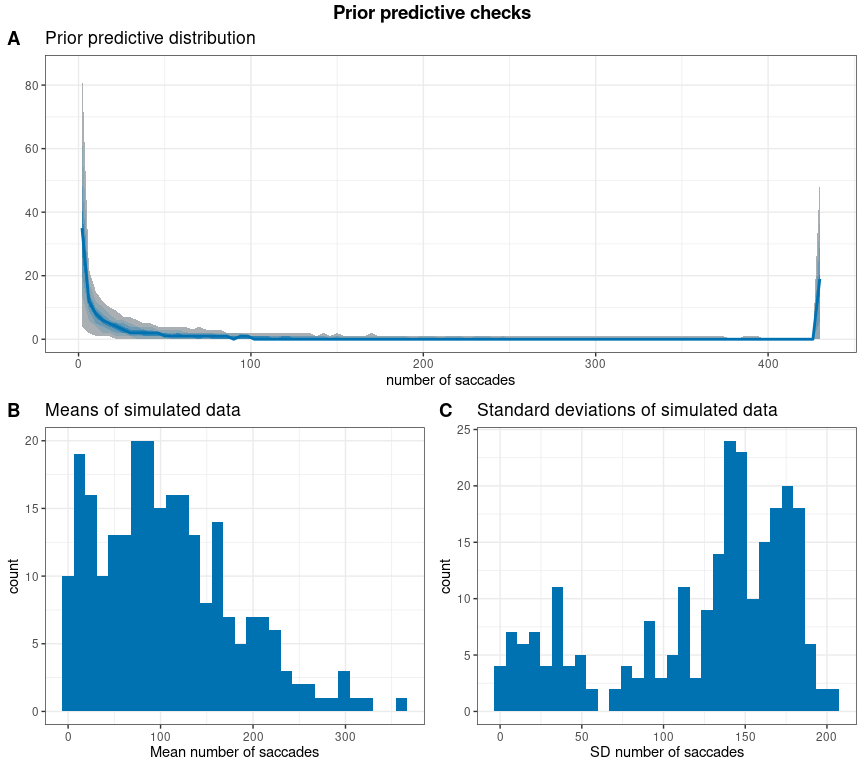
# check if the SBC already exists  
if (file.exists(file.path(cache\_dir, sprintf("df\_res\_%s.rds", code)))) {  
 # load in the results of the SBC  
 df.results = readRDS(file.path(cache\_dir, sprintf("df\_res\_%s.rds", code)))  
 df.backend = readRDS(file.path(cache\_dir, sprintf("df\_div\_%s.rds", 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.

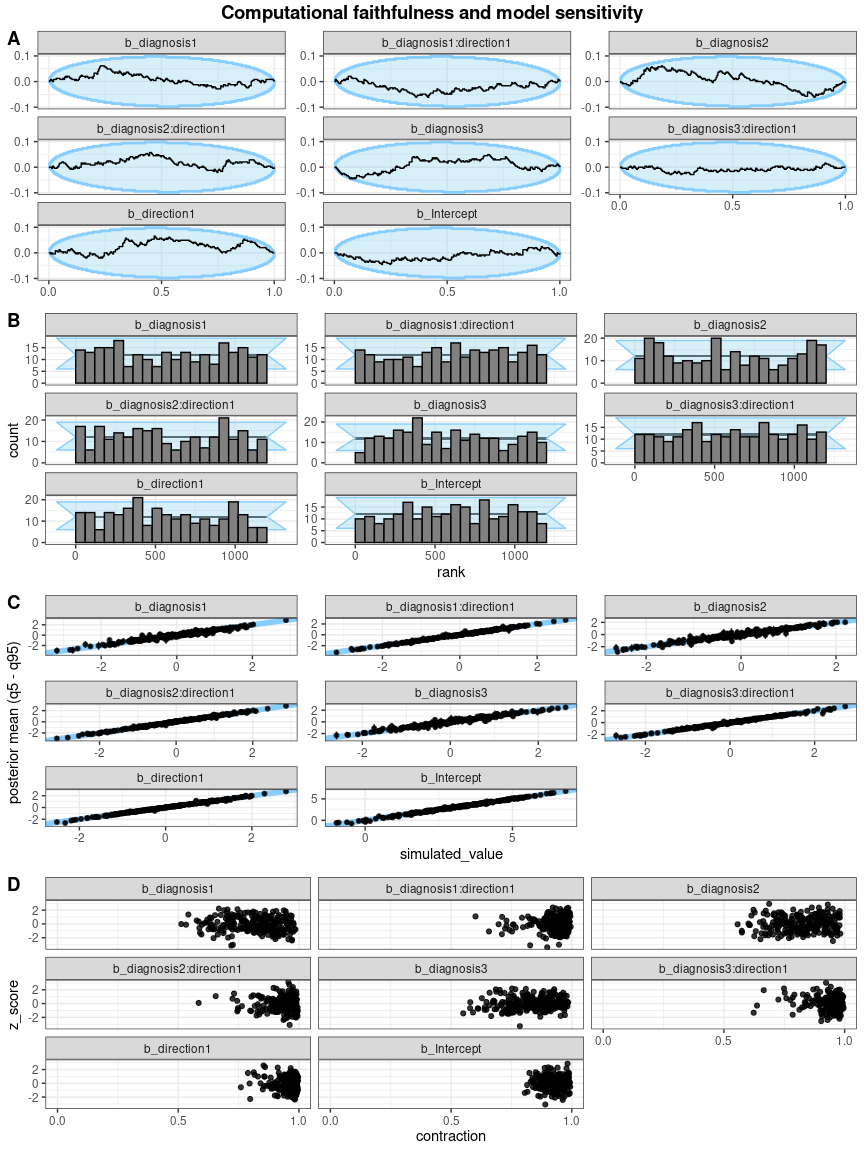
# get the true values  
truePars = dat[["variables"]]  
  
# create a matrix out of generated data  
dvname = gsub(" ", "", gsub("[\\|~].\*", "", f.cnt)[1])  
dvfakemat = matrix(NA, nrow(dat[['generated']][[1]]), length(dat[['generated']]))   
for (i in 1:length(dat[['generated']])) {  
 dvfakemat[,i]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

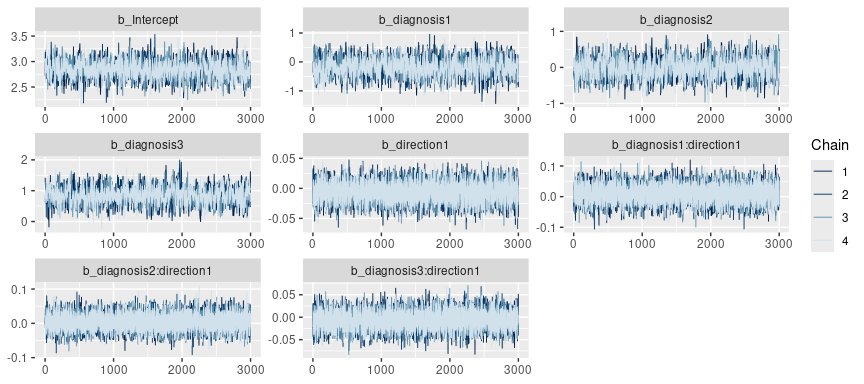
## E-BFMI indicated no pathological behavior.

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

## [1] 2

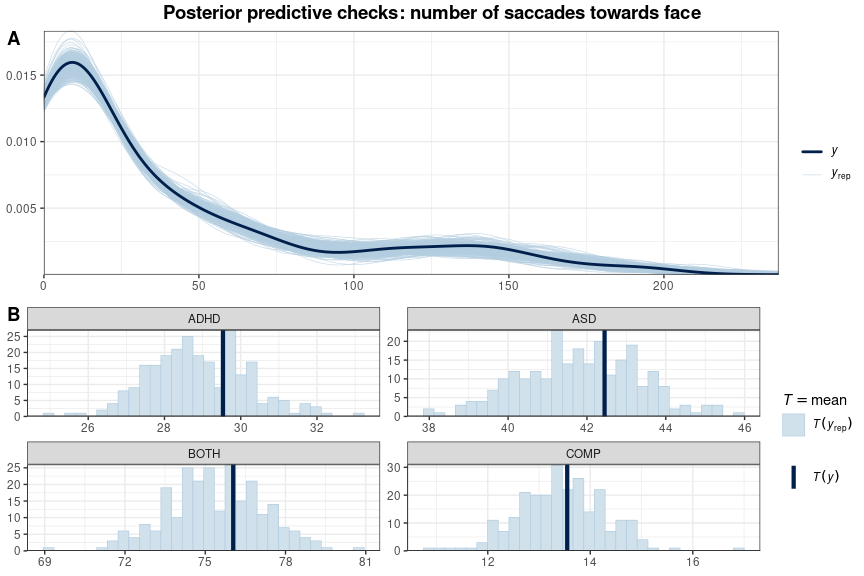
# check the trace plots  
post.draws = as\_draws\_df(m.cnt)  
mcmc\_trace(post.draws, regex\_pars = "^b\_",  
 facet\_args = list(ncol = 3)) +  
 scale\_x\_continuous(breaks=scales::pretty\_breaks(n = 3)) +  
 scale\_y\_continuous(breaks=scales::pretty\_breaks(n = 3))

## 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.

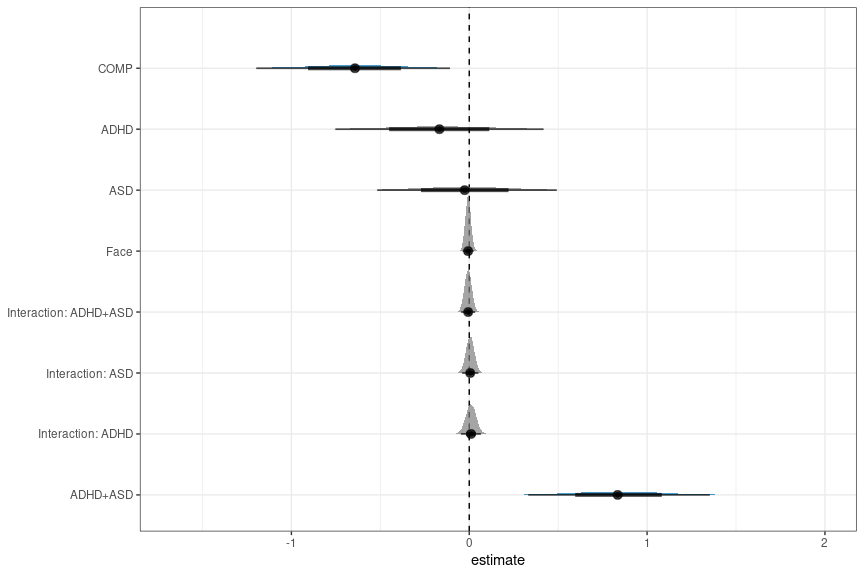
## Inferences

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).

# 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.

# extract predicted differences  
df.new = df.cnt %>%   
 select(diagnosis, direction) %>%   
 distinct() %>%  
 mutate(  
 condition = paste(diagnosis, direction, sep = "\_")  
 )  
df.ms = as.data.frame(  
 fitted(m.cnt, summary = F,   
 newdata = df.new %>% select(diagnosis, direction),   
 re\_formula = NA))  
colnames(df.ms) = df.new$condition  
  
st(df.ms,  
 summ = c('mean(x)','sd(x)','min(x)','pctile(x)[2.5]',  
 'pctile(x)[97.5]','max(x)'))

Summary Statistics

| Variable | Mean | Sd | Min | Pctile[2.5] | Pctile[97.5] | Max |
| --- | --- | --- | --- | --- | --- | --- |
| COMP\_face | 9.3 | 3.1 | 2.6 | 4.5 | 16 | 33 |
| COMP\_object | 9.6 | 3.1 | 2.7 | 4.7 | 17 | 29 |
| ADHD\_face | 15 | 5.7 | 3.3 | 6.9 | 28 | 62 |
| ADHD\_object | 15 | 5.6 | 3.5 | 6.8 | 28 | 62 |
| ASD\_face | 18 | 5.5 | 4.4 | 9.1 | 30 | 51 |
| ASD\_object | 18 | 5.5 | 4.3 | 9.1 | 30 | 51 |
| BOTH\_face | 41 | 12 | 9.6 | 21 | 68 | 126 |
| BOTH\_object | 42 | 13 | 9.7 | 22 | 70 | 126 |

st(df.ms %>%   
 mutate(  
 face = rowMeans(select(., matches(".\*\_face")), na.rm = T),  
 object = rowMeans(select(., matches(".\*\_object")), na.rm = T),  
 FAB = object - face  
 ) %>% select(face, object, FAB),  
 summ = c('mean(x)','sd(x)','min(x)','pctile(x)[2.5]','pctile(x)[97.5]','max(x)'))

Summary Statistics

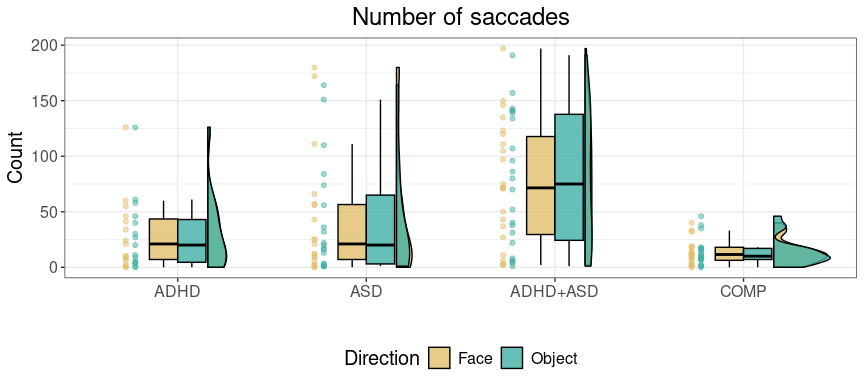
| Variable | Mean | Sd | Min | Pctile[2.5] | Pctile[97.5] | Max |
| --- | --- | --- | --- | --- | --- | --- |
| face | 21 | 3.9 | 11 | 14 | 29 | 45 |
| object | 21 | 3.9 | 11 | 14 | 30 | 44 |
| FAB | 0.32 | 0.58 | -2 | -0.86 | 1.4 | 3.2 |

Our hypothesis regarding the number of saccades towards face or object cues was not confirmed: there was no credible difference 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"),  
 Direction = recode(direction, "face" = "Face", "object" = "Object")  
 ) %>%  
 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 = "Count") +  
 theme\_bw() +   
 theme(legend.position = "bottom",   
 plot.title = element\_text(hjust = 0.5),   
 legend.direction = "horizontal",   
 text = element\_text(size = 15))



ggsave("Fig5\_nrSac.svg",   
 units = "mm",   
 width = 170,  
 height = 100,  
 dpi = 300)

# S2.3 Latencies of target-elicited 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

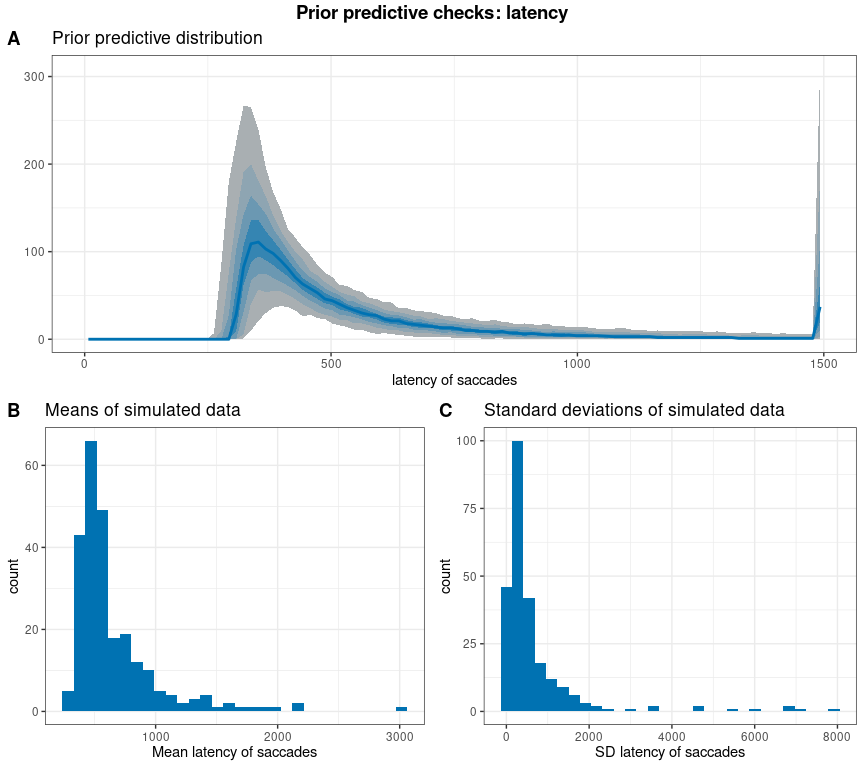
if (file.exists(file.path(cache\_dir, paste0("df\_res\_", code, ".rds")))) {  
 # load in the results of the SBC  
 df.results = readRDS(file.path(cache\_dir, paste0("df\_res\_", code, ".rds")))  
 df.backend = readRDS(file = file.path(cache\_dir, paste0("df\_div\_", code, ".rds")))  
 dat = readRDS(file = 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.

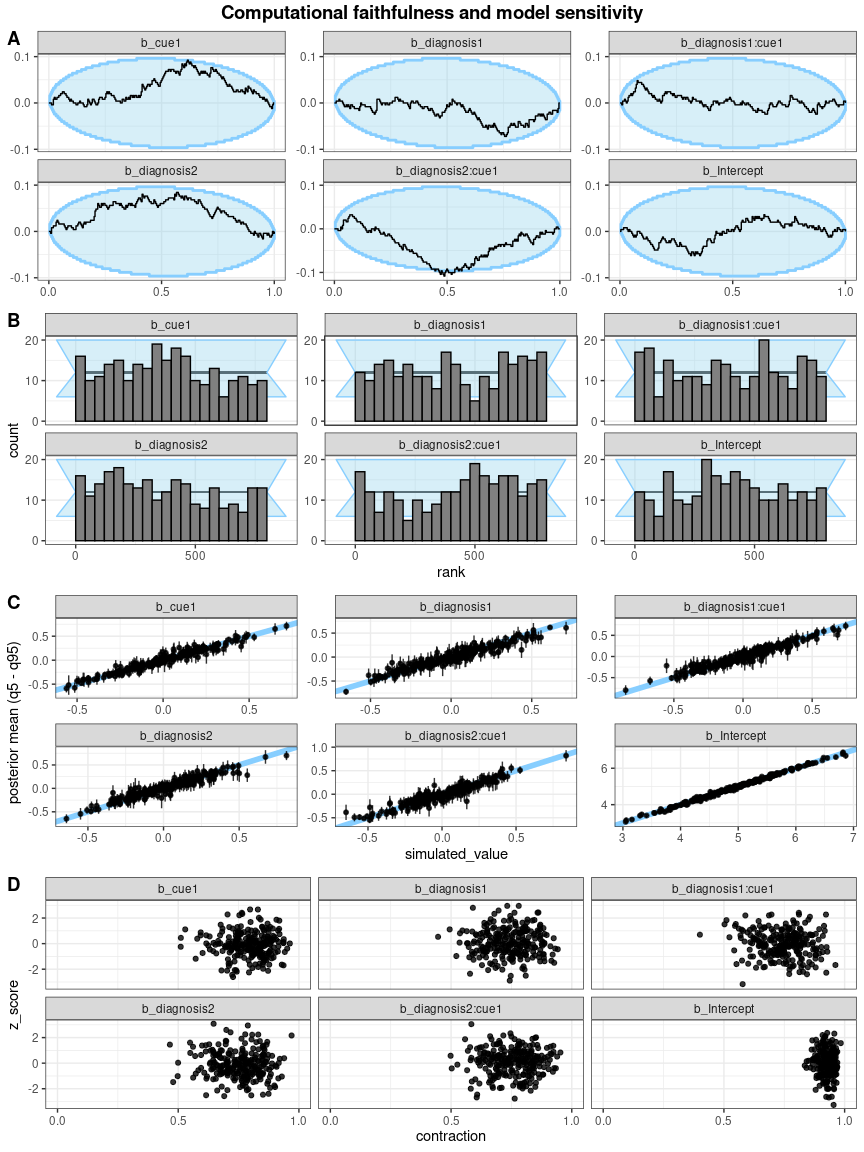
# get the true values  
truePars = dat[["variables"]]  
  
# create a matrix out of generated data  
dvname = gsub(" ", "", gsub("[\\|~].\*", "", f.lat)[1])  
dvfakemat = matrix(NA, nrow(dat[['generated']][[1]]), length(dat[['generated']]))   
for (i in 1:length(dat[['generated']])) {  
 dvfakemat[,i] = 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   
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$sim\_id)) %>%  
 ungroup() %>%  
 mutate(  
 max\_rank = max(rank)  
 )  
p1 = plot\_ecdf\_diff(df.results.b) + theme\_bw() + theme(legend.position = "none") +  
 scale\_x\_continuous(breaks=scales::pretty\_breaks(n = 3)) +  
 scale\_y\_continuous(breaks=scales::pretty\_breaks(n = 3))  
p2 = plot\_rank\_hist(df.results.b, bins = 20) + theme\_bw() +  
 scale\_x\_continuous(breaks=scales::pretty\_breaks(n = 3)) +  
 scale\_y\_continuous(breaks=scales::pretty\_breaks(n = 3))  
p3 = plot\_sim\_estimated(df.results.b, alpha = .8) + theme\_bw() +  
 scale\_x\_continuous(breaks=scales::pretty\_breaks(n = 3)) +  
 scale\_y\_continuous(breaks=scales::pretty\_breaks(n = 3))  
p4 = plot\_contraction(df.results.b,   
 prior\_sd = setNames(  
 c(as.numeric(  
 gsub(".\*, (.+)\\).\*", "\\1",   
 priors[priors$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:

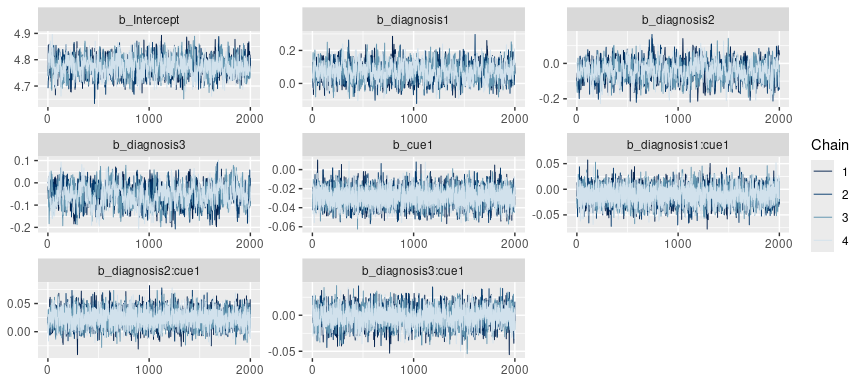
## E-BFMI indicated no pathological behavior.

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

## [1] 0

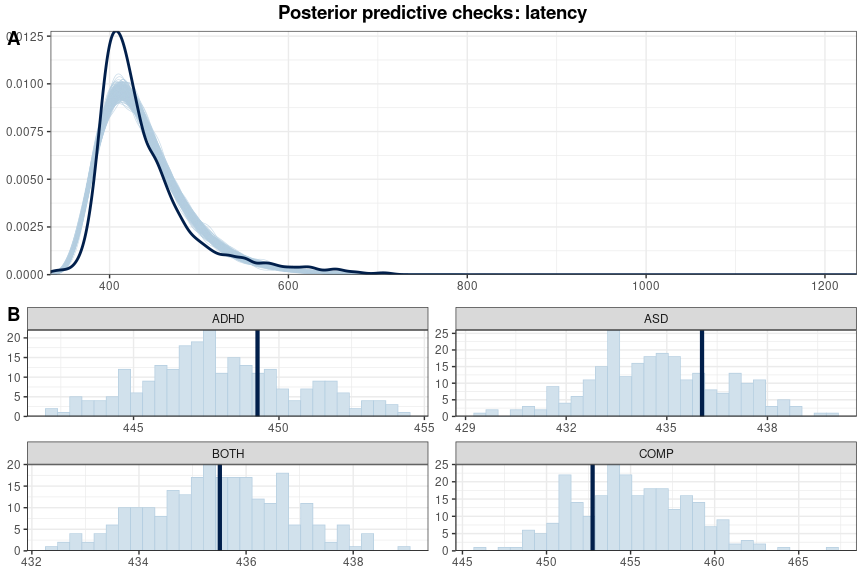
# check the trace plots  
post.draws = as\_draws\_df(m.lat)  
mcmc\_trace(post.draws, regex\_pars = "^b\_",   
 facet\_args = list(ncol = 3)) +  
 scale\_x\_continuous(breaks=scales::pretty\_breaks(n = 3)) +  
 scale\_y\_continuous(breaks=scales::pretty\_breaks(n = 3))

## 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.

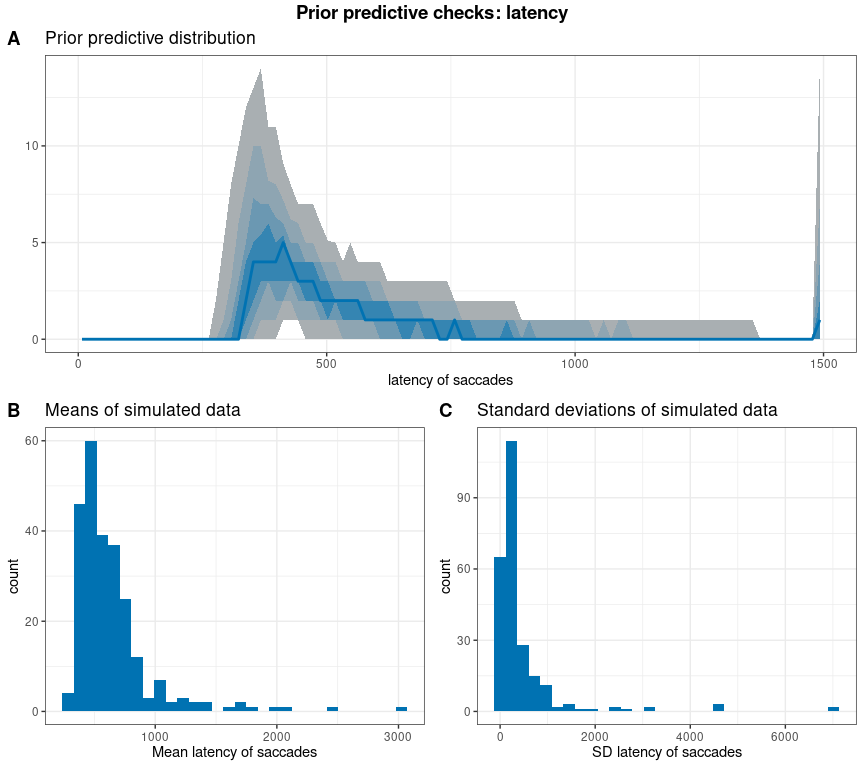
if (file.exists(file.path(cache\_dir, paste0("df\_res\_", code, ".rds")))) {  
 # load in the results of the SBC  
 df.results = readRDS(file.path(cache\_dir, paste0("df\_res\_", code, ".rds")))  
 df.backend = readRDS(file = file.path(cache\_dir, paste0("df\_div\_", code, ".rds")))  
 dat = readRDS(file = 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.

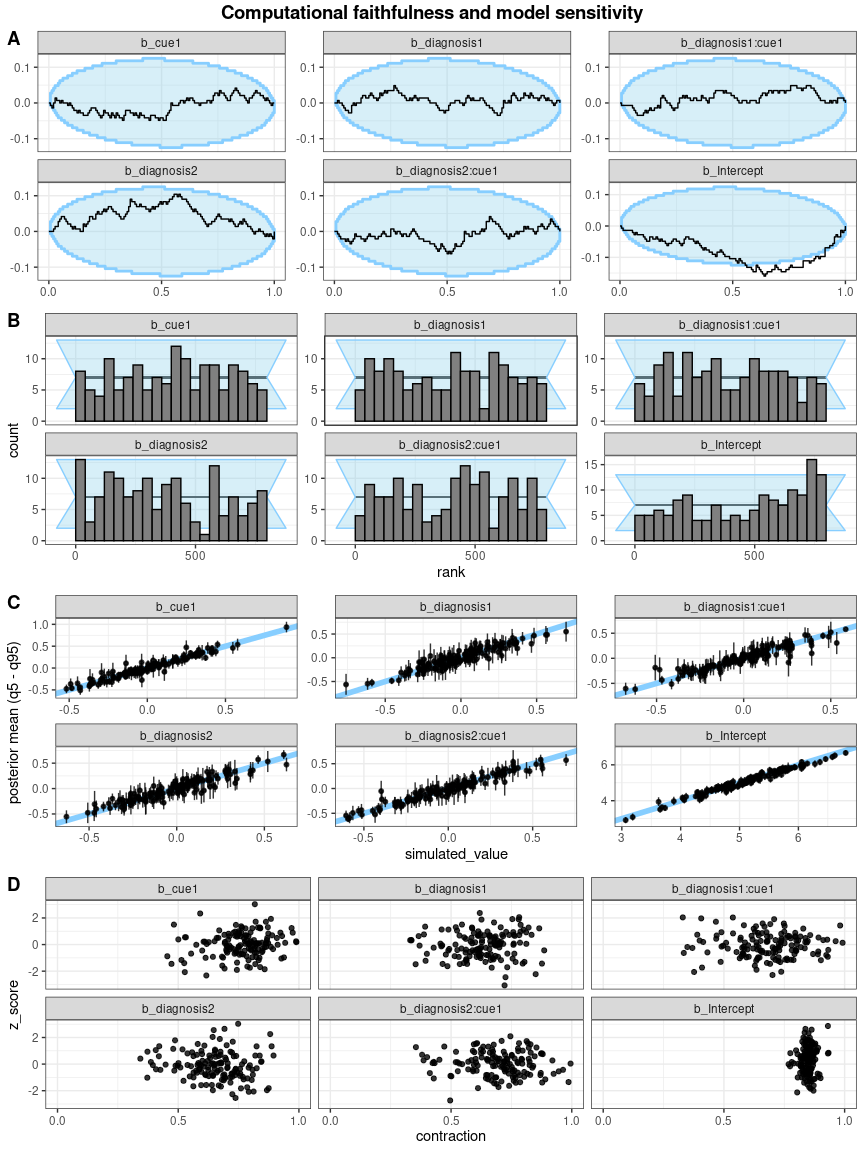
# get the true values  
truePars = dat[["variables"]]  
  
# create a matrix out of generated data  
dvname = gsub(" ", "", gsub("[\\|~].\*", "", f.lat)[1])  
dvfakemat = matrix(NA, nrow(dat[['generated']][[1]]), length(dat[['generated']]))   
for (i in 1:length(dat[['generated']])) {  
 dvfakemat[,i]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

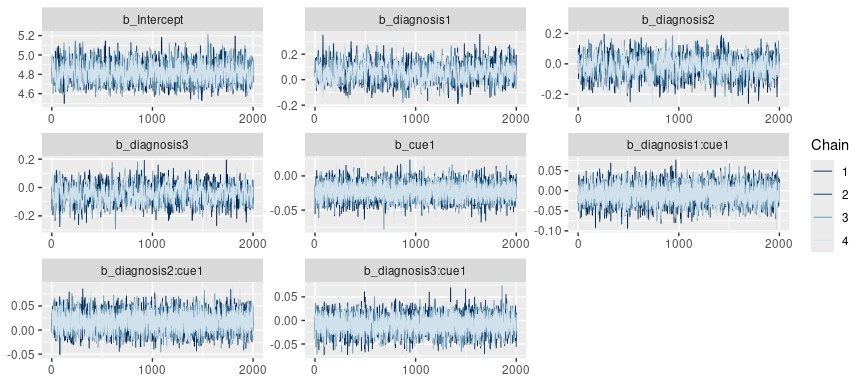
## E-BFMI indicated no pathological behavior.

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

## [1] 0

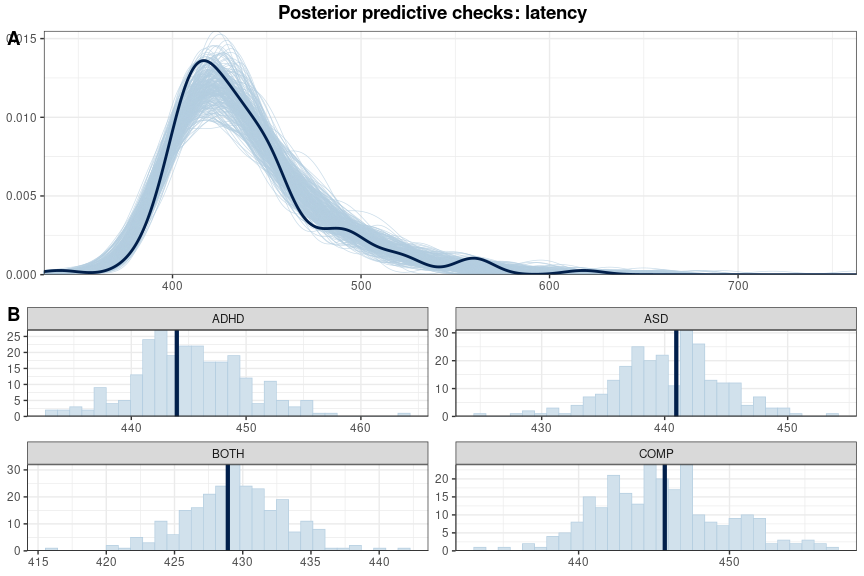
# check the trace plots  
post.draws = as\_draws\_df(m.lat)  
mcmc\_trace(post.draws, regex\_pars = "^b\_",   
 facet\_args = list(ncol = 3)) +  
 scale\_x\_continuous(breaks=scales::pretty\_breaks(n = 3)) +  
 scale\_y\_continuous(breaks=scales::pretty\_breaks(n = 3))

## 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.

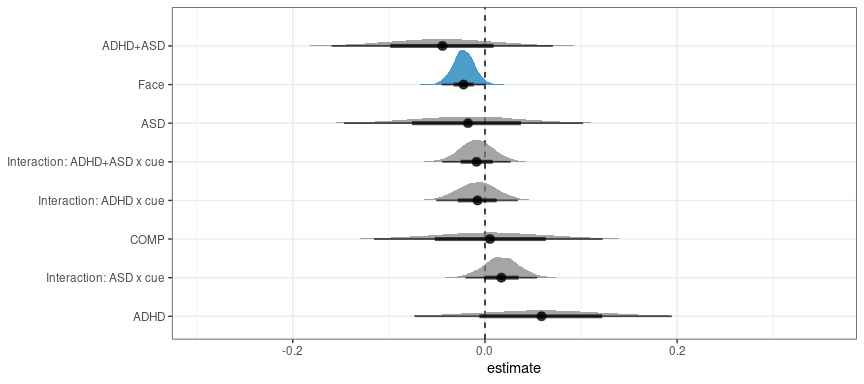
### Inferences

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).

# 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  
  
st(df.ms,  
 summ = c('mean(x)','sd(x)','min(x)','pctile(x)[2.5]',  
 'pctile(x)[97.5]','max(x)'))

Summary Statistics

| Variable | Mean | Sd | Min | Pctile[2.5] | Pctile[97.5] | Max |
| --- | --- | --- | --- | --- | --- | --- |
| COMP\_face | 432 | 8.9 | 405 | 415 | 450 | 475 |
| COMP\_object | 438 | 9.4 | 408 | 420 | 457 | 476 |
| ADHD\_face | 438 | 11 | 401 | 418 | 460 | 494 |
| ADHD\_object | 446 | 11 | 408 | 424 | 469 | 504 |
| ASD\_face | 432 | 9 | 402 | 414 | 449 | 465 |
| ASD\_object | 433 | 9.1 | 404 | 415 | 451 | 473 |
| BOTH\_face | 426 | 7.9 | 402 | 410 | 441 | 462 |
| BOTH\_object | 433 | 8.4 | 406 | 416 | 449 | 482 |

# calculate our difference columns  
df.ms = df.ms %>%  
 mutate(  
 e = rowMeans(select(., matches(".\*\_object")), na.rm = T) -   
 rowMeans(select(., matches(".\*\_face")), na.rm = T)  
 )  
  
st(df.ms %>%   
 mutate(  
 # get the face and object latencies -200, so they start with target onset  
 face = rowMeans(select(., matches(".\*\_face")), na.rm = T) - 200,  
 object = rowMeans(select(., matches(".\*\_object")), na.rm = T) - 200,  
 FAB = object - face  
 ) %>% select(face, object, FAB),  
 summ = c('mean(x)','sd(x)','min(x)','pctile(x)[2.5]','pctile(x)[97.5]','max(x)'))

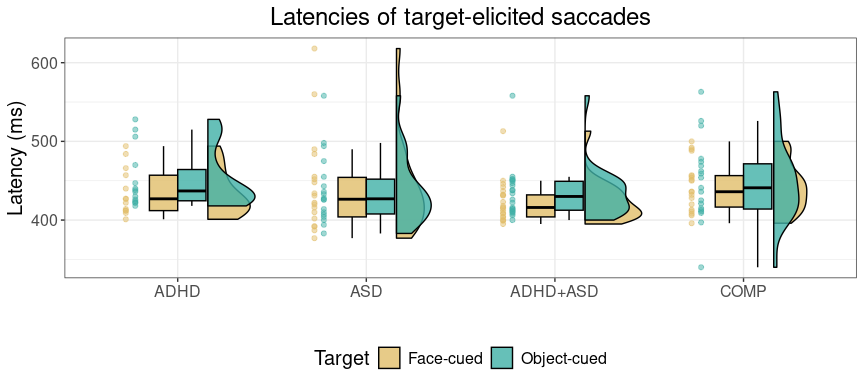
Summary Statistics

| Variable | Mean | Sd | Min | Pctile[2.5] | Pctile[97.5] | Max |
| --- | --- | --- | --- | --- | --- | --- |
| face | 232 | 4.7 | 217 | 222 | 241 | 252 |
| object | 238 | 4.9 | 222 | 228 | 247 | 262 |
| FAB | 5.6 | 2.8 | -5.9 | -0.27 | 11 | 17 |

Our hypothesis that target-elicited 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 that latencies of target-elicited saccades where faster in response to face-cued compared to object-cued targets regardless of the diagnostic group (*estimate* = 0.04 [0, 0.09], *posterior probability* = 97.52%). Specifically, the model predicted a 5.56ms [0.01, 11.18] shorter latency of for saccades produced towards a face-cued compared to an object-cued target.

### Plots

# rain cloud plot for the   
df.lat.agg %>%  
 mutate(  
 diagnosis = recode(diagnosis, "BOTH" = "ADHD+ASD"),  
 Target = recode(cue, "face" = "Face-cued", "object" = "Object-cued")  
 ) %>%   
 ggplot(aes(diagnosis, lat, fill = Target, colour = Target)) + #  
 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 target-elicited saccades", x = "", y = "Latency (ms)") +  
 theme\_bw() +   
 theme(legend.position = "bottom",   
 plot.title = element\_text(hjust = 0.5),   
 legend.direction = "horizontal",   
 text = element\_text(size = 15))

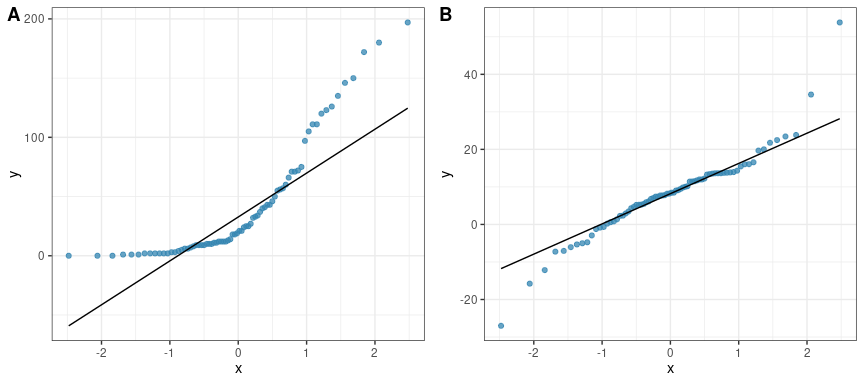


ggsave("Fig6\_latSac.svg",   
 units = "mm",   
 width = 170,  
 height = 100,  
 dpi = 300)

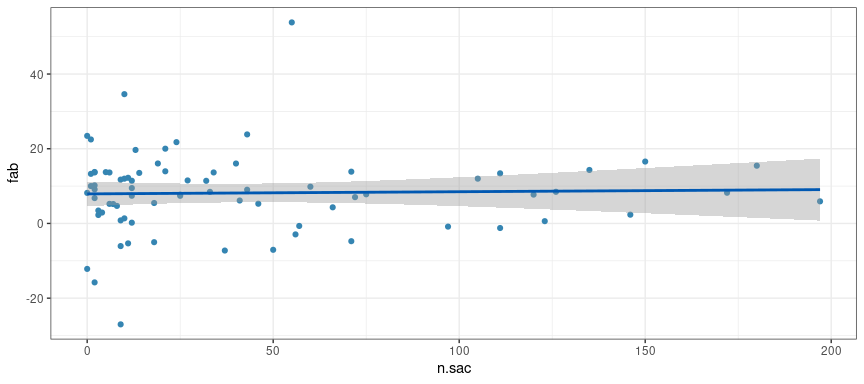
# S2.4 Correlation with reaction times: number of saccades

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 = c\_mid\_highlight) +   
 stat\_qq\_line() +   
 theme\_bw()  
p2 = ggplot(df.diff, aes(sample = fab)) +   
 stat\_qq(alpha = 0.75, colour = c\_mid\_highlight) +   
 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("rho\_CNT.rds")) {  
 rhoSamples.cnt = readRDS("rho\_CNT.rds")  
} else {  
 set.seed(1597)  
 rhoSamples.cnt =   
 spearmanGibbsSampler(xVals = df.diff$n.sac,  
 yVals = df.diff$fab,   
 nSamples = 5e3)  
 saveRDS(rhoSamples.cnt, file = "rho\_CNT.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 = c\_mid\_highlight) +  
 geom\_smooth(method = "lm",   
 formula = y ~ x,   
 geom = "smooth", colour = c\_dark\_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 moderate evidence against an association between the number of saccades and face attention bias (log(*BF*) = -1.98).

# S2.5 Exploration: cue-elicited 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. (2020)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-elicited 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:

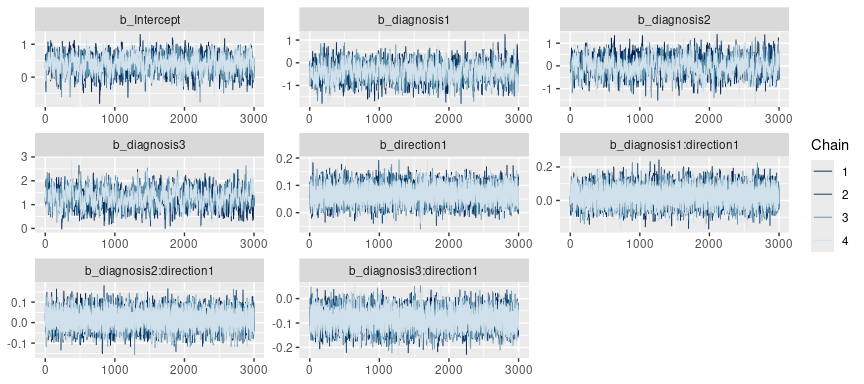
## E-BFMI indicated no pathological behavior.

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

## [1] 0

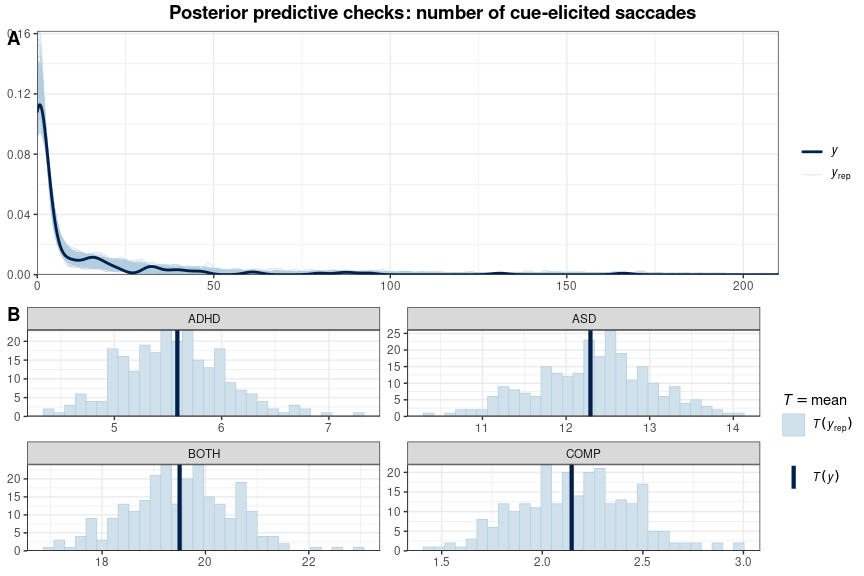
# check the trace plots  
post.draws = as\_draws\_df(m.cnt)  
mcmc\_trace(post.draws, regex\_pars = "^b\_",  
 facet\_args = list(ncol = 3)) +  
 scale\_x\_continuous(breaks=scales::pretty\_breaks(n = 3)) +  
 scale\_y\_continuous(breaks=scales::pretty\_breaks(n = 3))

## 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-elicited saccades",   
 face = "bold", size = 14))



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

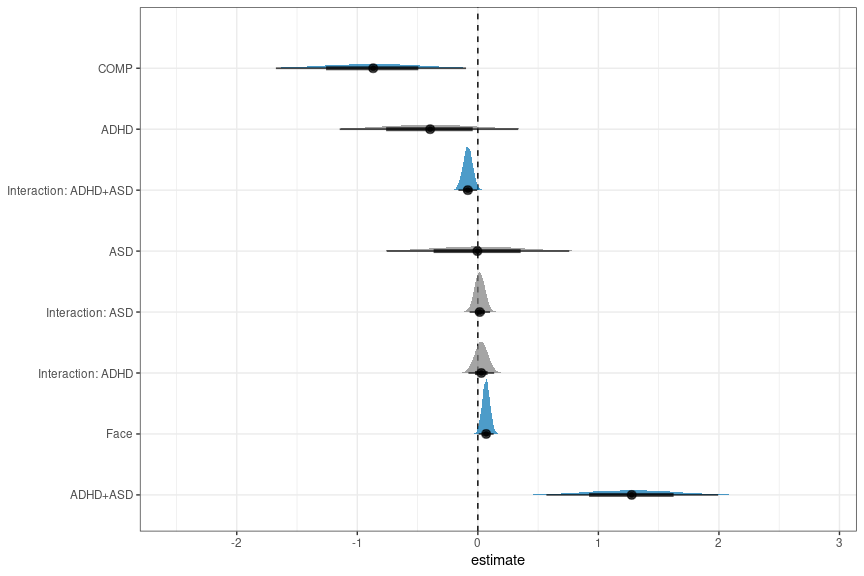
## Inferences

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).

# 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.

# extract predicted differences  
df.new = df.cnt.cue %>% ungroup() %>%  
 select(diagnosis, direction) %>%   
 distinct() %>%  
 mutate(  
 condition = paste(diagnosis, direction, sep = "\_")  
 )  
df.ms = as.data.frame(  
 fitted(m.cnt, summary = F,   
 newdata = df.new %>% select(diagnosis, direction),   
 re\_formula = NA))  
colnames(df.ms) = df.new$condition  
  
st(df.ms,  
 summ = c('mean(x)','sd(x)','min(x)','pctile(x)[2.5]',  
 'pctile(x)[97.5]','max(x)'))

Summary Statistics

| Variable | Mean | Sd | Min | Pctile[2.5] | Pctile[97.5] | Max |
| --- | --- | --- | --- | --- | --- | --- |
| ADHD\_face | 1.2 | 0.6 | 0.21 | 0.42 | 2.6 | 6.4 |
| ADHD\_object | 1 | 0.5 | 0.15 | 0.34 | 2.2 | 6.6 |
| COMP\_face | 0.78 | 0.4 | 0.099 | 0.25 | 1.7 | 4.6 |
| COMP\_object | 0.63 | 0.33 | 0.08 | 0.2 | 1.4 | 3.7 |
| ASD\_face | 1.8 | 0.85 | 0.23 | 0.63 | 3.8 | 8.8 |
| ASD\_object | 1.5 | 0.72 | 0.18 | 0.53 | 3.2 | 7.4 |
| BOTH\_face | 5.8 | 2.6 | 0.91 | 2.1 | 11 | 31 |
| BOTH\_object | 6 | 2.7 | 0.98 | 2.2 | 12 | 33 |

st(df.ms %>%   
 mutate(  
 face = rowMeans(select(., matches(".\*\_face")), na.rm = T),  
 object = rowMeans(select(., matches(".\*\_object")), na.rm = T),  
 FAB = object - face  
 ) %>% select(face, object, FAB),  
 summ = c('mean(x)','sd(x)','min(x)','pctile(x)[2.5]','pctile(x)[97.5]','max(x)'))

Summary Statistics

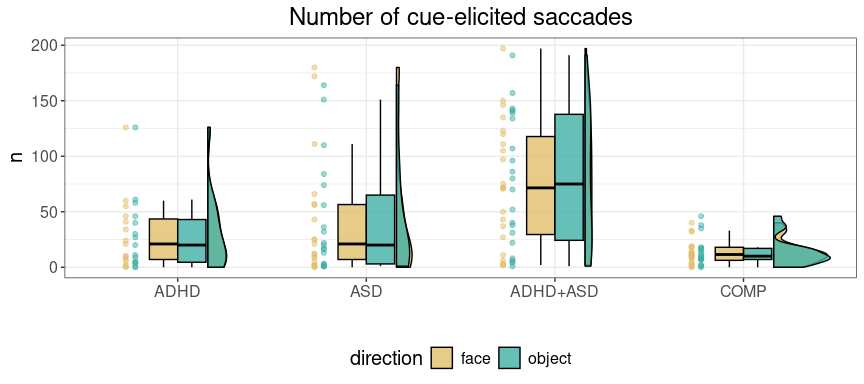
| Variable | Mean | Sd | Min | Pctile[2.5] | Pctile[97.5] | Max |
| --- | --- | --- | --- | --- | --- | --- |
| face | 2.4 | 0.73 | 0.63 | 1.3 | 4 | 8.7 |
| object | 2.3 | 0.72 | 0.56 | 1.2 | 3.8 | 8.8 |
| FAB | -0.12 | 0.14 | -0.77 | -0.42 | 0.14 | 0.77 |

On average, participants produced cue-elicited saccades on 4.55% +- 1.04 of the trials. However, the range was very wide, with some participants producing none and others producing them on 68.75% of the trials. Regardless of group, credibly more cue-elicited saccades were produced towards face compared to object cues (*estimate* = -0.14 [-0.26, -0.01], *posterior probability* = 98.54%).

## Plots

As a last step, we can 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 cue-elicited 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))



# S2.6 Exploration: Latencies of cue-elicited saccades

## Model

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-elicited 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:

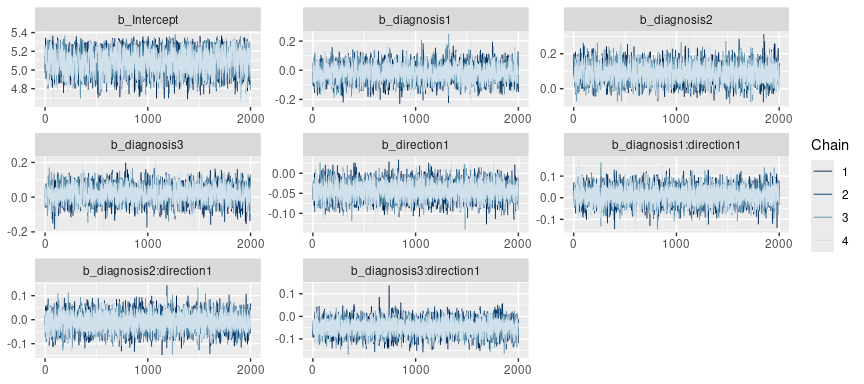
## E-BFMI indicated no pathological behavior.

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

## [1] 0

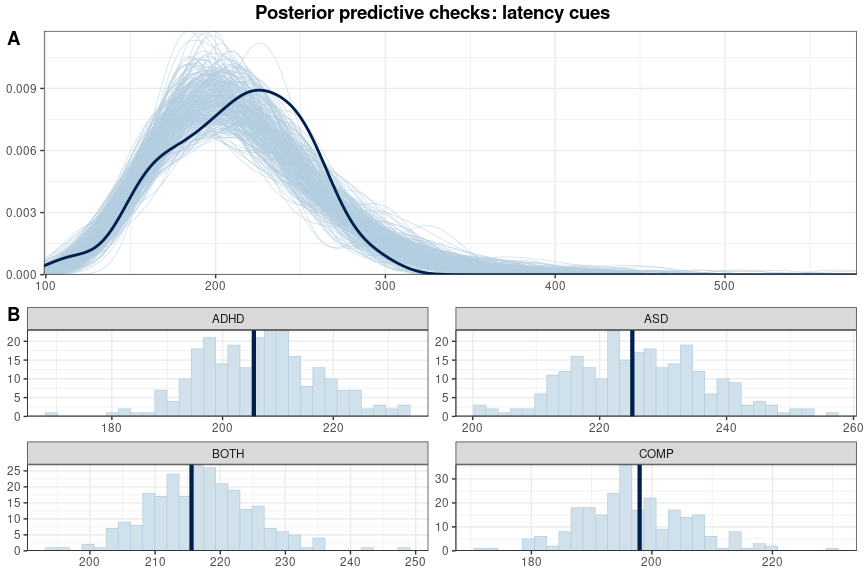
# check the trace plots  
post.draws = as\_draws\_df(m.lat)  
mcmc\_trace(post.draws, regex\_pars = "^b\_",   
 facet\_args = list(ncol = 3)) +  
 scale\_x\_continuous(breaks=scales::pretty\_breaks(n = 3)) +  
 scale\_y\_continuous(breaks=scales::pretty\_breaks(n = 3))

## 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.

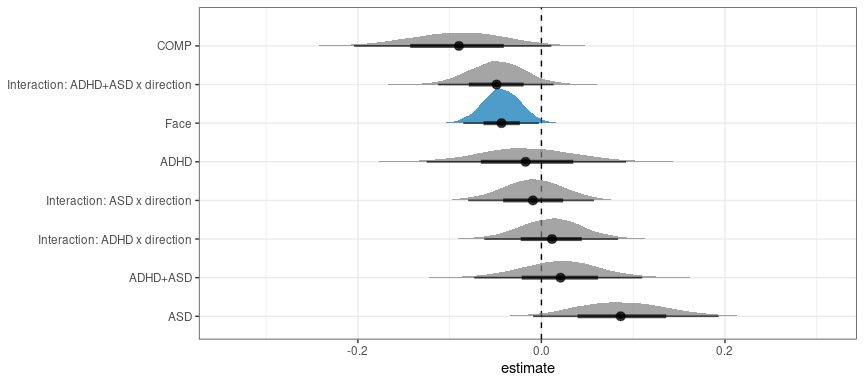
## Inferences

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).

# 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  
  
st(df.ms,  
 summ = c('mean(x)','sd(x)','min(x)','pctile(x)[2.5]',  
 'pctile(x)[97.5]','max(x)'))

Summary Statistics

| Variable | Mean | Sd | Min | Pctile[2.5] | Pctile[97.5] | Max |
| --- | --- | --- | --- | --- | --- | --- |
| ASD\_face | 214 | 12 | 169 | 190 | 238 | 272 |
| ADHD\_face | 200 | 12 | 154 | 178 | 224 | 252 |
| ADHD\_object | 211 | 13 | 170 | 185 | 237 | 263 |
| ASD\_object | 233 | 13 | 194 | 208 | 259 | 284 |
| COMP\_face | 194 | 11 | 158 | 173 | 217 | 240 |
| COMP\_object | 193 | 11 | 157 | 172 | 215 | 241 |
| BOTH\_face | 197 | 9.7 | 159 | 177 | 215 | 235 |
| BOTH\_object | 228 | 11 | 188 | 206 | 249 | 277 |

# calculate our difference columns  
df.ms = df.ms %>%  
 mutate(  
 e = rowMeans(select(., matches(".\*\_object")), na.rm = T) -   
 rowMeans(select(., matches(".\*\_face")), na.rm = T)  
 )  
  
st(df.ms %>%   
 mutate(  
 face = rowMeans(select(., matches(".\*\_face")), na.rm = T),  
 object = rowMeans(select(., matches(".\*\_object")), na.rm = T),  
 FAB = e  
 ) %>% select(face, object, FAB),  
 summ = c('mean(x)','sd(x)','min(x)','pctile(x)[2.5]','pctile(x)[97.5]','max(x)'))

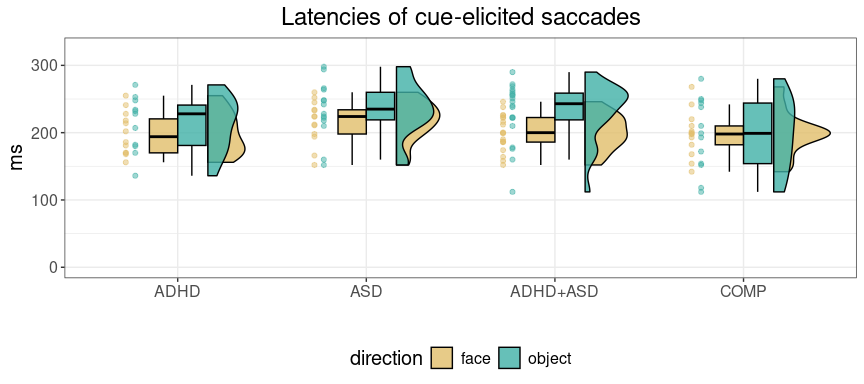
Summary Statistics

| Variable | Mean | Sd | Min | Pctile[2.5] | Pctile[97.5] | Max |
| --- | --- | --- | --- | --- | --- | --- |
| face | 201 | 6 | 179 | 190 | 213 | 225 |
| object | 216 | 6.3 | 193 | 204 | 229 | 241 |
| FAB | 15 | 6.8 | -9.7 | 0.9 | 27 | 41 |

A similar effect was found when exploring the latencies of cue-induced saccade, with participants producing saccades towards face cues faster than saccades towards object cues, independent of group (*estimate* = 0.09 [0.01, 0.17], *posterior probability* = 98.16%). This model predicted that if a cue-induced saccade was produced towards the face, the latency was 14.92ms [1.41, 27.87] shorter than if a saccade was produced towards 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 cue-elicited 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))



# S2.7 Exploration: Dwell times starting within cue-elicited saccade window

## Model

We assume that the SBC from the reaction time lognormal model holds here, although we slightly widen the Intercept because we have less prior knowledge.

# read in the preprocessed data  
df.fix = readRDS("FAB\_ET\_fix.rds")  
  
# set the formula  
f.fix = brms::bf(duration ~ diagnosis \* ROI \* onTar +   
 (ROI \* onTar | subID) + (ROI \* onTar | subID))  
  
# set weakly informative priors  
priors = c(  
 # general priors based on SBV  
 prior(normal(6, 0.6), class = Intercept),  
 prior(normal(0, 0.5), class = sigma),  
 prior(normal(0, 0.1), class = sd),  
 prior(lkj(2), class = cor),  
 prior(normal(0, 0.04), class = b),  
 # shift  
 prior(normal(200, 100), class = ndt)  
)  
  
# set number of iterations and warmup for models  
iter = 3000  
warm = 1000  
  
# set the contrasts  
contrasts(df.fix$ROI) = contr.sum(2)  
contrasts(df.fix$ROI)

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

contrasts(df.fix$onTar) = contr.sum(2)  
contrasts(df.fix$onTar)

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

contrasts(df.fix$diagnosis) = contr.sum(4)  
contrasts(df.fix$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(5599)  
m.fix = brm(f.fix,  
 df.fix, prior = priors,  
 iter = iter, warmup = warm,  
 backend = "cmdstanr", threads = threading(8),  
 family = "shifted\_lognormal",  
 file = "m\_fix",  
 save\_pars = save\_pars(all = TRUE)  
 )  
rstan::check\_hmc\_diagnostics(m.fix$fit)

##   
## Divergences:

## 0 of 8000 iterations ended with a divergence.

##   
## Tree depth:

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

##   
## Energy:

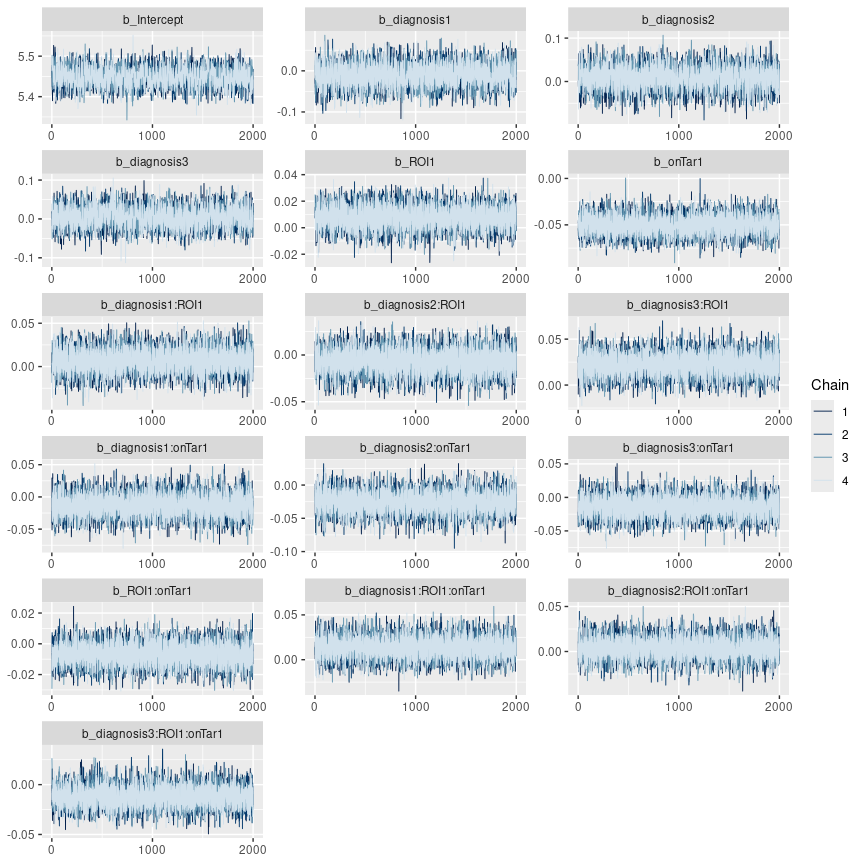
## E-BFMI indicated no pathological behavior.

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

## [1] 0

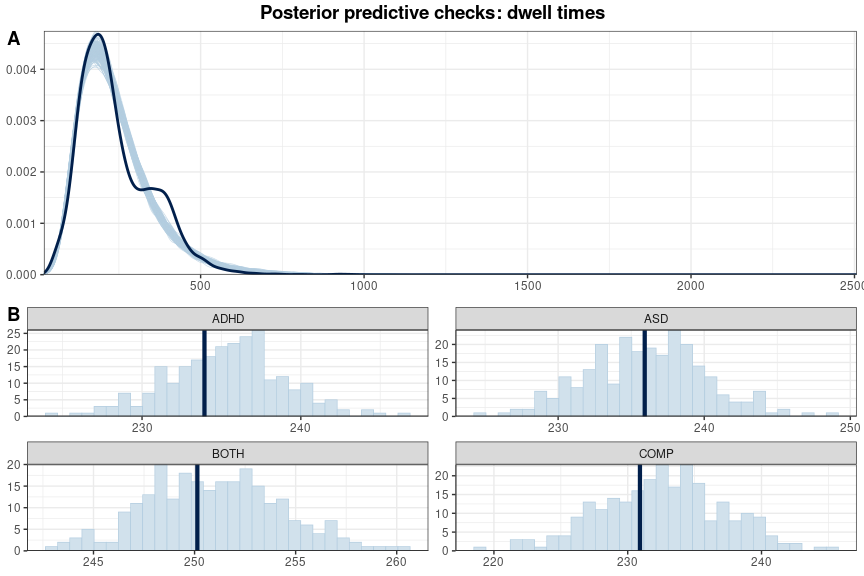
# check the trace plots  
post.draws = as\_draws\_df(m.fix)  
mcmc\_trace(post.draws, regex\_pars = "^b\_",   
 facet\_args = list(ncol = 3)) +  
 scale\_x\_continuous(breaks=scales::pretty\_breaks(n = 3)) +  
 scale\_y\_continuous(breaks=scales::pretty\_breaks(n = 3))

## 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.fix, ndraws = nsim)  
  
# check the fit of the predicted data compared to the real data  
p1 = pp\_check(m.fix, 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.fix$duration, post.pred, df.fix$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: dwell times",   
 face = "bold", size = 14))



This looks good.

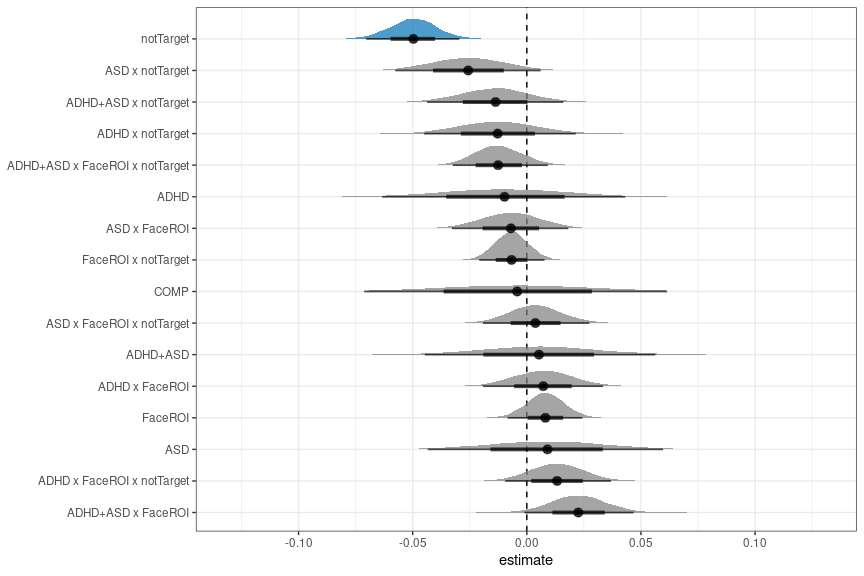
## Inferences

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.fix)

## Family: shifted\_lognormal   
## Links: mu = identity; sigma = identity; ndt = identity   
## Formula: duration ~ diagnosis \* ROI \* onTar + (ROI \* onTar | subID) + (ROI \* onTar | subID)   
## Data: df.fix (Number of observations: 5702)   
## Draws: 4 chains, each with iter = 3000; warmup = 1000; thin = 1;  
## total post-warmup draws = 8000  
##   
## Multilevel Hyperparameters:  
## ~subID (Number of levels: 81)   
## Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS  
## sd(Intercept) 0.18 0.02 0.15 0.22 1.00 3070  
## sd(ROI1) 0.04 0.01 0.02 0.06 1.00 2708  
## sd(onTar1) 0.07 0.01 0.05 0.09 1.00 3719  
## sd(ROI1:onTar1) 0.03 0.01 0.00 0.05 1.00 1855  
## cor(Intercept,ROI1) -0.29 0.18 -0.62 0.07 1.00 7330  
## cor(Intercept,onTar1) 0.19 0.15 -0.10 0.47 1.00 6795  
## cor(ROI1,onTar1) 0.15 0.22 -0.29 0.57 1.00 2071  
## cor(Intercept,ROI1:onTar1) -0.18 0.24 -0.62 0.30 1.00 10680  
## cor(ROI1,ROI1:onTar1) -0.12 0.30 -0.67 0.49 1.00 5538  
## cor(onTar1,ROI1:onTar1) -0.39 0.25 -0.81 0.18 1.00 6031  
## Tail\_ESS  
## sd(Intercept) 5144  
## sd(ROI1) 3108  
## sd(onTar1) 6380  
## sd(ROI1:onTar1) 2246  
## cor(Intercept,ROI1) 5963  
## cor(Intercept,onTar1) 6242  
## cor(ROI1,onTar1) 4089  
## cor(Intercept,ROI1:onTar1) 5995  
## cor(ROI1,ROI1:onTar1) 5601  
## cor(onTar1,ROI1:onTar1) 4879  
##   
## Regression Coefficients:  
## Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS  
## Intercept 5.45 0.02 5.40 5.49 1.00 2106  
## diagnosis1 -0.01 0.03 -0.06 0.04 1.00 3590  
## diagnosis2 0.01 0.03 -0.04 0.06 1.00 4466  
## diagnosis3 0.01 0.03 -0.04 0.06 1.00 3299  
## ROI1 0.01 0.01 -0.01 0.02 1.00 8887  
## onTar1 -0.05 0.01 -0.07 -0.03 1.00 7364  
## diagnosis1:ROI1 0.01 0.01 -0.02 0.03 1.00 9460  
## diagnosis2:ROI1 -0.01 0.01 -0.03 0.02 1.00 9140  
## diagnosis3:ROI1 0.02 0.01 -0.00 0.05 1.00 8638  
## diagnosis1:onTar1 -0.01 0.02 -0.04 0.02 1.00 7596  
## diagnosis2:onTar1 -0.03 0.02 -0.06 0.01 1.00 7836  
## diagnosis3:onTar1 -0.01 0.02 -0.04 0.02 1.00 7465  
## ROI1:onTar1 -0.01 0.01 -0.02 0.01 1.00 9906  
## diagnosis1:ROI1:onTar1 0.01 0.01 -0.01 0.04 1.00 10271  
## diagnosis2:ROI1:onTar1 0.00 0.01 -0.02 0.03 1.00 11833  
## diagnosis3:ROI1:onTar1 -0.01 0.01 -0.03 0.01 1.00 8728  
## Tail\_ESS  
## Intercept 3392  
## diagnosis1 4862  
## diagnosis2 5798  
## diagnosis3 4760  
## ROI1 6311  
## onTar1 6478  
## diagnosis1:ROI1 6778  
## diagnosis2:ROI1 6585  
## diagnosis3:ROI1 6048  
## diagnosis1:onTar1 6817  
## diagnosis2:onTar1 6222  
## diagnosis3:onTar1 5516  
## ROI1:onTar1 6471  
## diagnosis1:ROI1:onTar1 6584  
## diagnosis2:ROI1:onTar1 6776  
## diagnosis3:ROI1:onTar1 5718  
##   
## Further Distributional Parameters:  
## Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS Tail\_ESS  
## sigma 0.41 0.00 0.40 0.42 1.00 13344 5723  
## ndt 0.57 0.55 0.02 2.07 1.00 9209 5068  
##   
## 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.fix) %>%   
 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 = substr(coef, 3, nchar(coef)),  
 coef = str\_replace\_all(coef, ":", " x "),  
 coef = str\_replace\_all(coef, "ROI1", "FaceROI"),  
 coef = str\_replace\_all(coef, "onTar1", "notTarget"),  
 coef = str\_replace\_all(coef, "diagnosis1", "ADHD"),  
 coef = str\_replace\_all(coef, "diagnosis2", "ASD"),  
 coef = str\_replace\_all(coef, "diagnosis3", "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")



## Plots

# rain cloud plot for the   
df.fix %>%  
 group\_by(subID, diagnosis, ROI, onTar) %>%  
 summarise(  
 duration = median(duration, na.rm = T)  
 ) %>%  
 mutate(  
 diagnosis = recode(diagnosis, "BOTH" = "ADHD+ASD")  
 ) %>%   
 ggplot(aes(diagnosis, duration, fill = ROI, colour = ROI)) + #  
 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 = "Dwell times starting during cue-elicited saccades", x = "", y = "") +  
 theme\_bw() +   
 facet\_wrap(. ~ onTar) +  
 theme(legend.position = "bottom",   
 plot.title = element\_text(hjust = 0.5),   
 legend.direction = "horizontal",   
 text = element\_text(size = 15))

