S1: behavioural, model-agnostic analysis with brms

I. S. Plank

2025-01-31

S1.1 Introduction

This R Markdown script analyses data from the PAL (probabilistic associative learning) task of the EMBA project. The data was preprocessed before being read into this script.

In this task, participants view faces for which they have to judge whether the portrayed emotion is positive or negative. The faces are preceeded by a tone which is predictive of the valence of the following face. The visual angle of the faces was 3.46 degrees wide and 4.72 degrees high.

Some general settings

```
# number of simulations
nsim = 500

# set number of iterations and warmup for models
iter = 6000
warm = 1500

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

Package versions

The following packages are used in this RMarkdown file:

```
## [1] "R version 4.4.2 (2024-10-31)"
```

```
## [1] "knitr version 1.46"
## [1] "ggplot2 version 3.5.1"
## [1] "brms version 2.21.0"
## [1] "designr version 0.1.13"
## [1] "bridgesampling version 1.1.2"
## [1] "tidyverse version 2.0.0"
## [1] "ggpubr version 0.6.0"
## [1] "ggrain version 0.0.4"
## [1] "bayesplot version 1.11.1"
## [1] "SBC version 0.3.0.9000"
## [1] "rstatix version 0.7.2"
## [1] "BayesFactor version 0.9.12.4.7"
## [1] "bayestestR version 0.15.0"
```

Introduction

We planned to determine the group-level effect subjects following Barr (2013). For each model, experiment specific priors were set based on previous literature or the task (see comments in the code).

We perform prior predictive checks as proposed in Schad, Betancourt and Vasishth (2020) using the SBC package. To do so, we create 500 simulated datasets where parameters are simulated from the priors. These parameters are used to create one fake dataset. Both the true underlying parameters and the simulated discrimination values are saved.

Then, we create graphs showing the prior predictive distribution of the simulated discrimination threshold to check whether our priors fit our general expectations about the data. Next, we perform checks of computational faithfulness and model sensitivity as proposed by Schad, Betancourt and Vasishth (2020) and implemented in the SBC package. We create models for each of the simulated datasets. Last, we calculate performance metrics for each of these models, focusing on the population-level parameters.

Preparation

First, we load the data and combine it with demographic information including the diagnostic status of the subjects. Then, all predictors are set to sum contrasts.

```
# check if the data file exists, if yes load it:
if (!file.exists("PAL_data.RData")) {

  # get demo info for subjects
  df.sub = read_csv(file.path("/home/emba/Documents/EMBA/CentraXX", "EMBA_centraXX.csv"),
                    show_col_types = F) %>%
    mutate(
      diagnosis = recode(diagnosis, "CTR" = "COMP")
    )

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

  # load the data: df.tsk, df.var
  load(file.path(dt.path, "df_PAL.RData"))

  # load excluded participants (accuracy < 2/3)
  exc = scan(file.path(dt.path, 'PAL_exc.txt'), what="character", sep=NULL)
  df.exc = df.sub %>% filter(subID %in% exc) %>%
    select(diagnosis) %>%
    group_by(diagnosis) %>% count()

  # merge with group
  df.tsk = merge(df.sub %>% select(subID, diagnosis),
                 df.tsk, all.y = T) %>%
    mutate_if(is.character, as.factor)
  df.var = merge(df.sub %>% select(subID, diagnosis),
                 df.var, all.y = T) %>%
    mutate_if(is.character, as.factor)

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

```
  ) %>%
  group_by(subID) %>%
  summarise(
    EDT = mean(EDT)
  )

# only keep participants included in the study in the subject data frame
df.sub = merge(df.tsk %>% select(subID) %>% distinct(), df.sub, all.x = T) %>%
  # merge with EDT dataframe
  merge(., df.fer, all.x = T)

# [!MISSING: load the eye tracking data]

# anonymise the data
df.tsk = df.tsk %>%
  mutate(
    PID = subID,
    subID = as.factor(as.numeric(subID))
  )

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

# anonymise other data in the same way
df.var$subID = str_replace_all(df.var$subID, recode)

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

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

# most of the measures are not normally distributed;
# therefore, we compute ranks for these outcomes
df.sub = df.sub %>%
  mutate(
    rage   = rank(age),
```

```
    rASRS  = rank(ASRS_total),
    rBDI   = rank(BDI_total),
    rRAADS = rank(RAADS_total),
    rTAS   = rank(TAS_total),
    diagnosis = as.factor(diagnosis)
  )

df.edt = df.sub %>% select(subID, diagnosis, EDT) %>% drop_na()

# now we can compute our ANOVAs
aov.age   = anovaBF(rage   ~ diagnosis, data = df.sub)
aov.iq    = anovaBF(iq     ~ diagnosis, data = df.sub)
aov.BDI   = anovaBF(rBDI   ~ diagnosis, data = df.sub)
aov.ASRS  = anovaBF(rASRS  ~ diagnosis, data = df.sub)
aov.RAADS = anovaBF(rRAADS ~ diagnosis, data = df.sub)
aov.TAS   = anovaBF(rTAS   ~ diagnosis, data = df.sub)

# ...and put everything in a new dataframe for printing
measurement  = "Age"
ADHD     = sprintf("%.2f (±%.2f)",
                   mean(df.sub[df.sub$diagnosis == "ADHD",]$age),
                   sd(df.sub[df.sub$diagnosis == "ADHD",]$age)/
                     sqrt(nrow(df.sub[df.sub$diagnosis == "ADHD",])))
ASD      = sprintf("%.2f (±%.2f)",
                   mean(df.sub[df.sub$diagnosis == "ASD",]$age),
                   sd(df.sub[df.sub$diagnosis == "ASD",]$age)/
                     sqrt(nrow(df.sub[df.sub$diagnosis == "ASD",])))
COMP      = sprintf("%.2f (±%.2f)",
                    mean(df.sub[df.sub$diagnosis == "COMP",]$age),
                    sd(df.sub[df.sub$diagnosis == "COMP",]$age)/
                      sqrt(nrow(df.sub[df.sub$diagnosis == "COMP",])))
logBF10 = sprintf("%.3f", aov.age@bayesFactor[["bf"]])
df.table = data.frame(measurement, ADHD, ASD, COMP, logBF10)
df.table = rbind(df.table,
     c(
       "ASRS",
       sprintf("%.2f (±%.2f)",
               mean(df.sub[df.sub$diagnosis == "ADHD",]$ASRS_total),
               sd(df.sub[df.sub$diagnosis == "ADHD",]$ASRS_total)/
                 sqrt(nrow(df.sub[df.sub$diagnosis == "ADHD",]))),
       sprintf("%.2f (±%.2f)",
               mean(df.sub[df.sub$diagnosis == "ASD",]$ASRS_total),
               sd(df.sub[df.sub$diagnosis == "ASD",]$ASRS_total)/
                 sqrt(nrow(df.sub[df.sub$diagnosis == "ASD",]))),
       sprintf("%.2f (±%.2f)",
               mean(df.sub[df.sub$diagnosis == "COMP",]$ASRS_total),
               sd(df.sub[df.sub$diagnosis == "COMP",]$ASRS_total)/
                 sqrt(nrow(df.sub[df.sub$diagnosis == "COMP",]))),
       sprintf("%.3f", aov.ASRS@bayesFactor[["bf"]])
     ),
     c(
       "BDI",
       sprintf("%.2f (±%.2f)",
```

```
            mean(df.sub[df.sub$diagnosis == "ADHD",]$BDI_total),
            sd(df.sub[df.sub$diagnosis == "ADHD",]$BDI_total)/
              sqrt(nrow(df.sub[df.sub$diagnosis == "ADHD",]))),
    sprintf("%.2f (±%.2f)",
            mean(df.sub[df.sub$diagnosis == "ASD",]$BDI_total),
            sd(df.sub[df.sub$diagnosis == "ASD",]$BDI_total)/
              sqrt(nrow(df.sub[df.sub$diagnosis == "ASD",]))),
    sprintf("%.2f (±%.2f)",
            mean(df.sub[df.sub$diagnosis == "COMP",]$BDI_total),
            sd(df.sub[df.sub$diagnosis == "COMP",]$BDI_total)/
              sqrt(nrow(df.sub[df.sub$diagnosis == "COMP",]))),
    sprintf("%.3f", aov.BDI@bayesFactor[["bf"]])
  ),
  c(
    "Gender (diverse/agender/non-binary - female - male)",
    sprintf("%d - %d - %d",
            nrow(df.sub[df.sub$diagnosis == "ADHD" & df.sub$gender == "dan",]),
            nrow(df.sub[df.sub$diagnosis == "ADHD" & df.sub$gender == "fem",]),
            nrow(df.sub[df.sub$diagnosis == "ADHD" & df.sub$gender == "mal",])),
    sprintf("%d - %d - %d",
            nrow(df.sub[df.sub$diagnosis == "ASD" & df.sub$gender == "dan",]),
            nrow(df.sub[df.sub$diagnosis == "ASD" & df.sub$gender == "fem",]),
            nrow(df.sub[df.sub$diagnosis == "ASD" & df.sub$gender == "mal",])),
    sprintf("%d - %d - %d",
            nrow(df.sub[df.sub$diagnosis == "COMP" & df.sub$gender == "dan",]),
            nrow(df.sub[df.sub$diagnosis == "COMP" & df.sub$gender == "fem",]),
            nrow(df.sub[df.sub$diagnosis == "COMP" & df.sub$gender == "mal",])),
    sprintf("%.3f", ct.full@bayesFactor[["bf"]])
  ),
  c(
    "IQ",
    sprintf("%.2f (±%.2f)",
            mean(df.sub[df.sub$diagnosis == "ADHD",]$iq),
            sd(df.sub[df.sub$diagnosis == "ADHD",]$iq)/
              sqrt(nrow(df.sub[df.sub$diagnosis == "ADHD",]))),
    sprintf("%.2f (±%.2f)",
            mean(df.sub[df.sub$diagnosis == "ASD",]$iq),
            sd(df.sub[df.sub$diagnosis == "ASD",]$iq)/
              sqrt(nrow(df.sub[df.sub$diagnosis == "ASD",]))),
    sprintf("%.2f (±%.2f)",
            mean(df.sub[df.sub$diagnosis == "COMP",]$iq),
            sd(df.sub[df.sub$diagnosis == "COMP",]$iq)/
              sqrt(nrow(df.sub[df.sub$diagnosis == "COMP",]))),
    sprintf("%.3f", aov.iq@bayesFactor[["bf"]])
  ),
  c(
    "RAADS",
    sprintf("%.2f (±%.2f)",
            mean(df.sub[df.sub$diagnosis == "ADHD",]$RAADS_total),
            sd(df.sub[df.sub$diagnosis == "ADHD",]$RAADS_total)/
              sqrt(nrow(df.sub[df.sub$diagnosis == "ADHD",]))),
    sprintf("%.2f (±%.2f)",
            mean(df.sub[df.sub$diagnosis == "ASD",]$RAADS_total),
```

```
                sd(df.sub[df.sub$diagnosis == "ASD",]$RAADS_total)/
                  sqrt(nrow(df.sub[df.sub$diagnosis == "ASD",]))),
        sprintf("%.2f (±%.2f)",
                mean(df.sub[df.sub$diagnosis == "COMP",]$RAADS_total),
                sd(df.sub[df.sub$diagnosis == "COMP",]$RAADS_total)/
                  sqrt(nrow(df.sub[df.sub$diagnosis == "COMP",]))),
        sprintf("%.3f", aov.RAADS@bayesFactor[["bf"]])
      ),
      c(
        "TAS",
        sprintf("%.2f (±%.2f)",
                mean(df.sub[df.sub$diagnosis == "ADHD",]$TAS_total),
                sd(df.sub[df.sub$diagnosis == "ADHD",]$TAS_total)/
                  sqrt(nrow(df.sub[df.sub$diagnosis == "ADHD",]))),
        sprintf("%.2f (±%.2f)",
                mean(df.sub[df.sub$diagnosis == "ASD",]$TAS_total),
                sd(df.sub[df.sub$diagnosis == "ASD",]$TAS_total)/
                  sqrt(nrow(df.sub[df.sub$diagnosis == "ASD",]))),
        sprintf("%.2f (±%.2f)",
                mean(df.sub[df.sub$diagnosis == "COMP",]$TAS_total),
                sd(df.sub[df.sub$diagnosis == "COMP",]$TAS_total)/
                  sqrt(nrow(df.sub[df.sub$diagnosis == "COMP",]))),
        sprintf("%.3f", aov.TAS@bayesFactor[["bf"]])
      ),
      c(
        "EDT",
        sprintf("%.2f (±%.2f)",
                mean(df.edt[df.edt$diagnosis == "ADHD",]$EDT),
                sd(df.edt[df.edt$diagnosis == "ADHD",]$EDT)/
                  sqrt(nrow(df.edt[df.edt$diagnosis == "ADHD",]))),
        sprintf("%.2f (±%.2f)",
                mean(df.edt[df.edt$diagnosis == "ASD",]$EDT),
                sd(df.edt[df.edt$diagnosis == "ASD",]$EDT)/
                  sqrt(nrow(df.edt[df.edt$diagnosis == "ASD",]))),
        sprintf("%.2f (±%.2f)",
                mean(df.edt[df.edt$diagnosis == "COMP",]$EDT),
                sd(df.edt[df.edt$diagnosis == "COMP",]$EDT)/
                  sqrt(nrow(df.edt[df.edt$diagnosis == "COMP",]))),
        sprintf("%.3f", 2.846)
      )
) %>% arrange(measurement)

# we aggregate the correct and the usable reaction times due to suboptimal
# posterior predictive fit of the full model
df.pal = df.tsk %>%
  group_by(subID, diagnosis, phase, expected, difficulty) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T),
    rt.use = median(rt.use, na.rm = T),
    acc    = 100 * mean(acc) # accuracy in percent
  )

# we also aggregate it to check whether there were any differences between
```

```
  # tone_pic combinations:
  df.tp = df.tsk %>%
    filter(expected != "neutral") %>% droplevels() %>%
    mutate(
      tone_pic = if_else(ut == 1, "highpos_lowneg", "highneg_lowpos")
    ) %>%
    group_by(subID, diagnosis, tone_pic, expected) %>%
    summarise(
      rt.cor = median(rt.cor, na.rm = T)
    )

  # save it all
  save(df.pal, df.var, df.tp, df.table, df.sht, ct.full, ct.mf, df.exc,
       file = "PAL_data.RData")

} else {

  load("PAL_data.RData")

}

# print the group of excluded participants
kable(df.exc)
```

diagnosis	n
ADHD	2
ASD	2
COMP	2

```
rm(df.exc)

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

diagnosis	variable	statistic	p	sig
ADHD	ASRS_total	0.9364986	0.1675120	
ASD	ASRS_total	0.9307625	0.1135925	
COMP	ASRS_total	0.9068414	0.0408291	*
ADHD	BDI_total	0.8019340	0.0005365	*
ASD	BDI_total	0.8511614	0.0028558	*
COMP	BDI_total	0.7469562	0.0000825	*
ADHD	EDT	0.9341534	0.1497323	
ASD	EDT	0.9727806	0.7744817	
COMP	EDT	0.9647553	0.5908591	
ADHD	RAADS_total	0.9270567	0.1065426	
ASD	RAADS_total	0.9556216	0.3808761	
COMP	RAADS_total	0.8644129	0.0061719	*
ADHD	TAS_total	0.9626930	0.5455600	
ASD	TAS_total	0.9240264	0.0812916	
COMP	TAS_total	0.8719909	0.0085270	*
ADHD	age	0.9439542	0.2386038	

diagnosis	variable	statistic	p	sig
ASD	age	0.9396864	0.1769487	
COMP	age	0.8212328	0.0010965	*
ADHD	iq	0.9736733	0.7942630	
ASD	iq	0.9591210	0.4458484	
COMP	iq	0.9589488	0.4683259	

```
rm(df.sht)

# print the outcome of the two contingency tables for comparison
ct.full@bayesFactor
```

```
##                          bf error                      time         code
## Non-indep. (a=1) -3.783988     0 Fri Jan 17 09:31:43 2025 fc8f76a509da
```

```
ct.mf@bayesFactor
```

```
##                          bf error                      time         code
## Non-indep. (a=1) -2.027069     0 Fri Jan 17 09:31:43 2025 fc8f7a7fac11
```

```
# drop the neutral condition for the analysis
df.pal = df.pal %>%
  filter(expected != "neutral") %>% droplevels() %>%
  mutate_if(is.character, as.factor) %>%
  ungroup()
df.var = df.var %>%
  filter(expected != "neutral") %>% droplevels() %>%
  mutate_if(is.character, as.factor) %>%
  ungroup()

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

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

```
contrasts(df.pal$expected) = contr.sum(2)
contrasts(df.pal$expected)
```

```
##            [,1]
## expected      1
## unexpected   -1
```

```
contrasts(df.pal$phase) = contr.sum(3)
contrasts(df.pal$phase)
```

```
##              [,1] [,2]
## prevolatile     1    0
## volatile        0    1
## postvolatile   -1   -1
```

```
contrasts(df.pal$difficulty) = contr.sum(3)
contrasts(df.pal$difficulty)
```

```
##           [,1] [,2]
## easy         1    0
## medium       0    1
## difficult   -1   -1
```

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

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

```
contrasts(df.var$expected) = contr.sum(2)
contrasts(df.var$expected)
```

```
##            [,1]
## expected      1
## unexpected   -1
```

```
contrasts(df.var$phase) = contr.sum(3)
contrasts(df.var$phase)
```

```
##              [,1] [,2]
## prevolatile     1    0
## volatile        0    1
## postvolatile   -1   -1
```

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

measurement	ADHD	ASD	COMP	logBF10
ASRS	43.95 (±2.59)	32.39 (±1.71)	25.95 (±1.70)	9.529
Age	26.23 (±1.41)	29.35 (±1.64)	27.59 (±1.25)	-1.408
BDI	7.95 (±1.78)	10.22 (±1.74)	2.32 (±0.68)	7.177
EDT	0.75 (±0.03)	0.80 (±0.02)	0.68 (±0.02)	2.846
Gender (diverse/agender/non-binary - female - male)	2 - 9 - 11	0 - 12 - 11	0 - 10 - 12	-3.784
IQ	108.57 (±2.39)	113.33 (±3.18)	109.14 (±1.84)	-1.293
RAADS	93.95 (±8.56)	150.26 (±8.37)	46.50 (±7.68)	21.151
TAS	50.82 (±2.49)	62.61 (±1.31)	39.36 (±2.07)	18.221

The three diagnostic groups are similar in age, IQ and gender distribution. However, they seem to differ in their questionnaire scores measuring ADHD (ASRS), depression (BDI), autism (RAADS) and alexithymia (TAS).

S1.2 Reaction time variance

In the preregistration, we noted the following population-level effects for the model of the reaction time variances: group, expectancy, phase and difficulty. However, the posterior fit for this model was suboptimal, therefore, we dropped the predictor difficulty.

Model setup

```
# figure out slopes for subjects
kable(head(df.var %>% count(subID, expected)))
```

subID	expected	n
1	expected	3
1	unexpected	3
10	expected	3
10	unexpected	3
11	expected	3
11	unexpected	3

```
kable(head(df.var %>% count(subID, phase)))
```

subID	phase	n
1	prevolatile	2
1	volatile	2
1	postvolatile	2
10	prevolatile	2
10	volatile	2
10	postvolatile	2

```
kable(head(df.var %>% count(subID, expected, phase)))
```

subID	expected	phase	n
1	expected	prevolatile	1
1	expected	volatile	1
1	expected	postvolatile	1
1	unexpected	prevolatile	1
1	unexpected	volatile	1
1	unexpected	postvolatile	1

```
# code for filenames
code = "PAL-var"

# model formula
f.var = brms::bf(rt.var ~ diagnosis * expected * phase +
                   (expected + phase | subID)
                 )

# set weakly informative priors
priors = c(
  prior(normal(4.50, 0.50),  class = Intercept),
  prior(normal(0,    0.50),  class = sigma),
  prior(normal(0.15, 0.15),  class = sd),
  prior(normal(0,    0.25),  class = b)
)
```

Simulation-based calibration

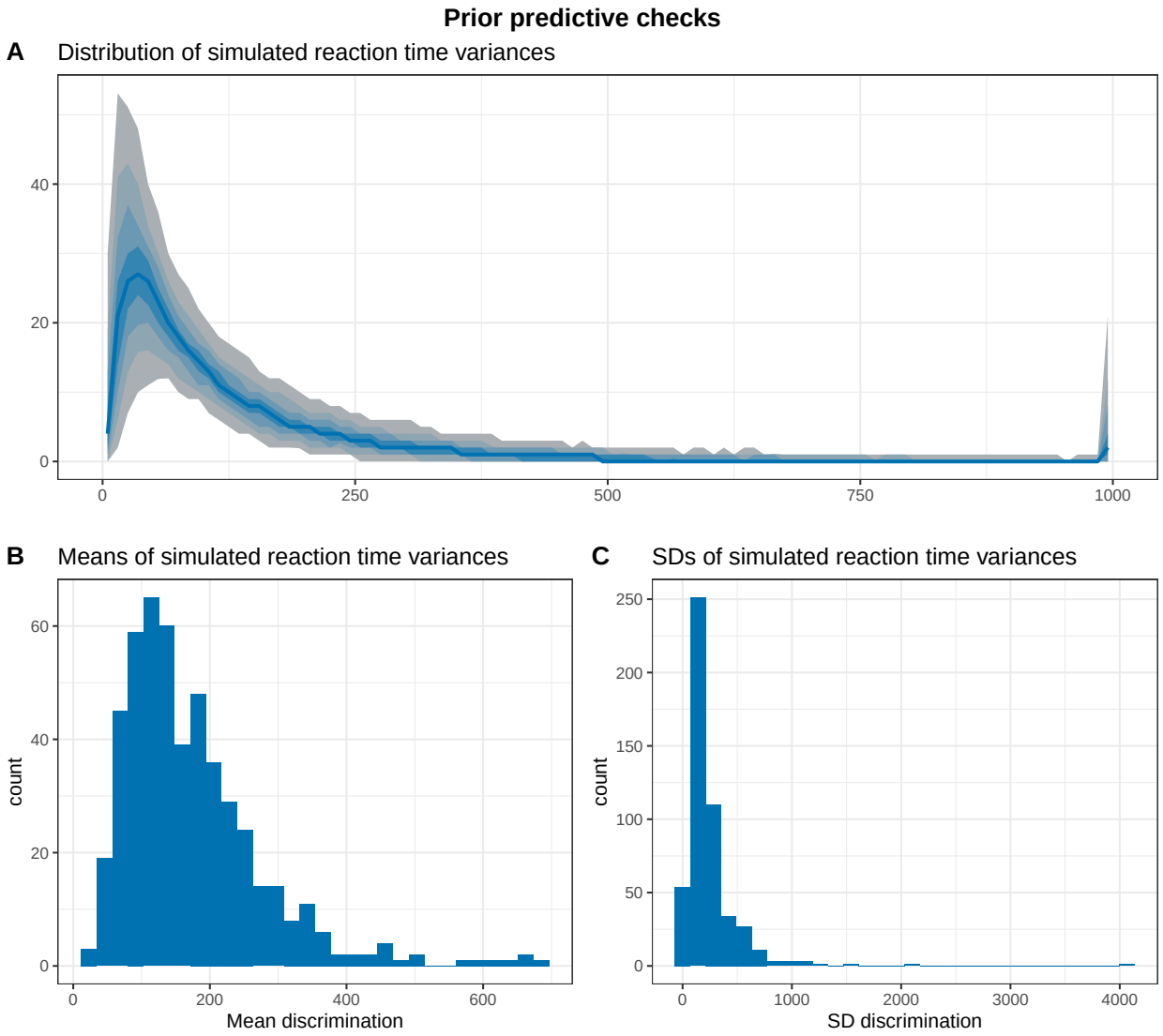
```
# check if the SBC already exists
if (file.exists(file.path(cache_dir, sprintf("df_res_%s.rds", code)))) {
  # load in the results of the SBC
  df.results = readRDS(file.path(cache_dir, sprintf("df_res_%s.rds", code)))
  df.backend = readRDS(file.path(cache_dir, sprintf("df_div_%s.rds", code)))
} else {
  # stimulate some data
  set.seed(2486)
  gen = SBC_generator_brms(f.var, data = df.var, prior = priors,
   thin = 50, warmup = 10000, refresh = 2000,
   generate_lp = TRUE, family = lognormal, 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)))
  # perform the SBC on the first 250 simulated datasets
  res = compute_SBC(SBC_datasets(dat$variables[1:250,],
                        dat$generated[1:250]),
        bck,
        cache_mode     = "results",
        cache_location = file.path(cache_dir, sprintf("res_%s", code)))
  saveRDS(res$stats, file = file.path(cache_dir, paste0("df_res_", code, ".rds")))
  saveRDS(res$backend_diagnostics, file = file.path(cache_dir, paste0("df_div_", code, ".rds")))
}
```

We start by investigating the rhats and the number of divergent samples. This shows that only 0 of 500 simulations had at least one parameter that had an rhat of at least 1.05 and 5 models had divergent samples. This suggests that this model performs well.

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

```
if (!(file.exists(file.path(cache_dir, sprintf("dvfakemat_%s", code))))) {
  # create a matrix out of generated data
  dvname = gsub(" ", "", gsub("[\\|~].*", "", f.var)[1])
  dvfakemat = matrix(NA, nrow(dat[['generated']][[1]]), length(dat[['generated']]))
  for (i in 1:length(dat[['generated']])) {
    dvfakemat[,i] = dat[['generated']][[i]][[dvname]]
  }
  saveRDS(dvfakemat, file.path(cache_dir, sprintf("dvfakemat_%s", code)))
} else {
  dvfakemat = readRDS(file.path(cache_dir, sprintf("dvfakemat_%s", code)))
}

# compute one histogram per simulated data-set
dvfakematH = dvfakemat
dvfakematH[dvfakematH > 1000] = 1000
breaks = seq(0, max(dvfakematH, na.rm=T), length.out = 101)
binwidth = breaks[2] - breaks[1]
histmat = matrix(NA, ncol = dim(dvfakemat)[2], nrow = length(breaks)-1)
for (i in 1:dim(dvfakemat)[2]) {
  histmat[,i] = hist(dvfakematH[,i], breaks = breaks, plot = F)$counts
}
# for each bin, compute quantiles across histograms
```

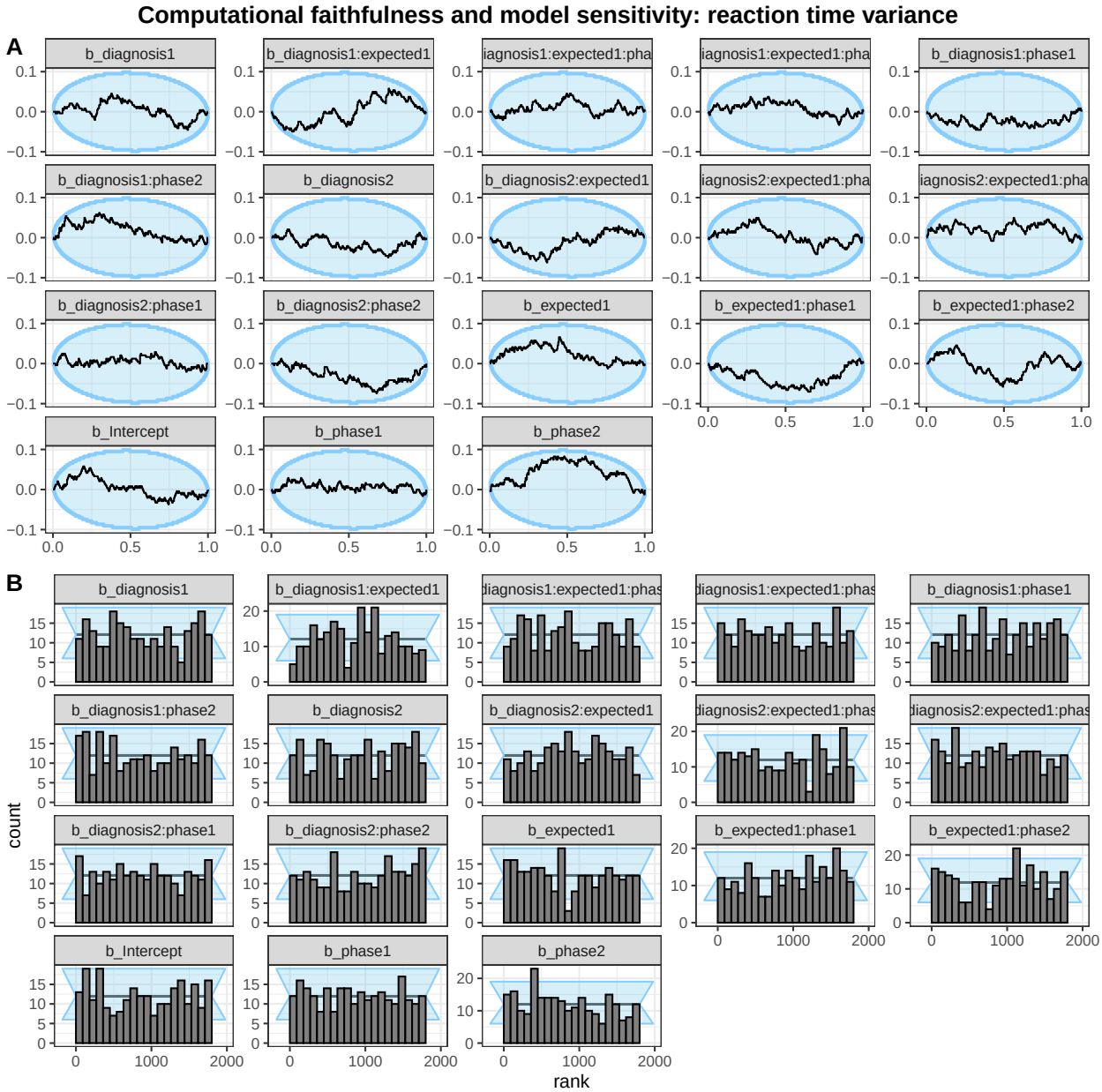
```r
probs = seq(0.1, 0.9, 0.1)
quantmat = as.data.frame(matrix(NA, nrow=dim(histmat)[1], ncol = length(probs)))
names(quantmat) = paste0("p", probs)
for (i in 1:dim(histmat)[1]) {
  quantmat[i,] = quantile(histmat[i,], p = probs)
}
quantmat$x = breaks[2:length(breaks)] - binwidth/2 # add bin mean
p1 = ggplot(data = quantmat, aes(x = x)) +
  geom_ribbon(aes(ymax = p0.9, ymin = p0.1), fill = c_light) +
  geom_ribbon(aes(ymax = p0.8, ymin = p0.2), fill = c_light_highlight) +
  geom_ribbon(aes(ymax = p0.7, ymin = p0.3), fill = c_mid) +
  geom_ribbon(aes(ymax = p0.6, ymin = p0.4), fill = c_mid_highlight) +
  geom_line(aes(y = p0.5), colour = c_dark, linewidth = 1) +
  labs(title = "Distribution of simulated reaction time variances", y = "", x = "") +
  theme_bw()

tmpM = apply(dvfakemat, 2, mean) # mean
tmpSD = apply(dvfakemat, 2, sd)
p2 = ggplot() +
  stat_bin(aes(x = tmpM), fill = c_dark)  +
  labs(x = "Mean discrimination", title = "Means of simulated reaction time variances") +
  theme_bw()
p3 = ggplot() +
  stat_bin(aes(x = tmpSD), fill = c_dark) +
  labs(x = "SD discrimination", title = "SDs of simulated reaction time variances") +
  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))
```



Subfigure B shows the distribution of the simulated data with bluer bands being more likely than greyer bands. It shows a general distribution that fits our expectations, even though there are quite a few values that are unreastically large. This is because we had to increase the standard deviations in the priors to achieve appropriate contraction values. The same applies to the distribution of the means and standard deviations in the simulated datasets. We go ahead with these priors and check the results of the SBC. We only plot the results from the models that had no divergence issues.

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

# plot SBC with functions from the SBC package focusing on population-level parameters
a = 0.5
df.results.b = df.results %>%
```

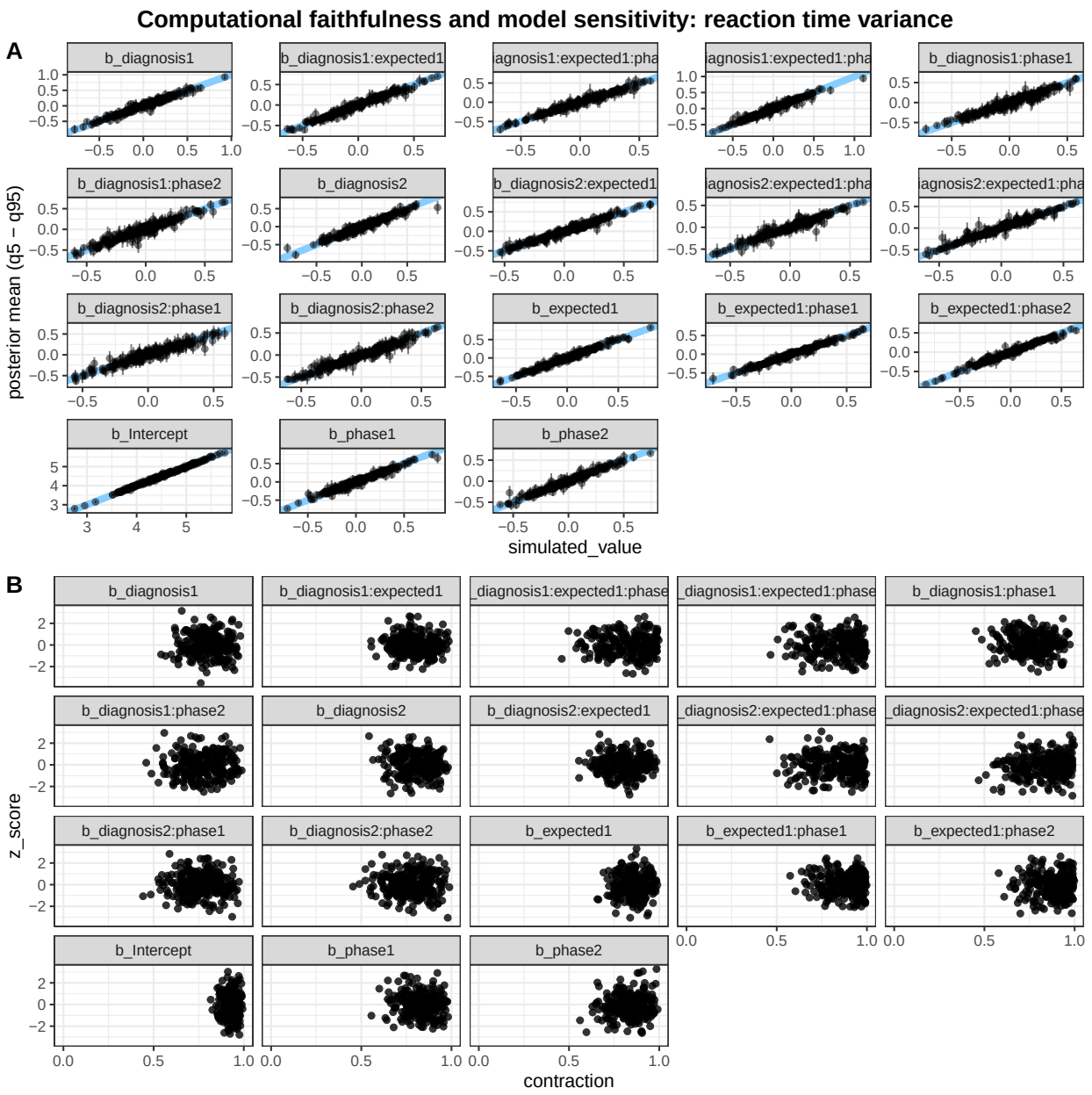
```
  filter(substr(variable, 1, 2) == "b_") %>%
  filter(!(sim_id %in% check$sim_id))
p1 = plot_ecdf_diff(df.results.b) + theme_bw() + theme(legend.position = "none") +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p2 = plot_rank_hist(df.results.b, bins = 20) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))

p = ggarrange(p1, p2, labels = "AUTO", ncol = 1, nrow = 2)
annotate_figure(p, top = text_grob("Computational faithfulness and model sensitivity: reaction time var
```



Next, we check the ranks of the parameters. If the model is unbiased, these should be uniformly distributed

(Schad, Betancourt and Vasishth, 2020). The sample empirical cumulative distribution function (ECDF) shows deviations from the theoretical distribution (95%) and the rank histogram also shows ranks outside the 95% expected range. However, we judge this to be acceptable as we cannot identify clear bias patterns as described in the information accompanying the SBC package: https://hyunjimoon.github.io/SBC/articles/rank_visualizations.html

```
p3 = plot_sim_estimated(df.results.b, alpha = a) + 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(0.50, rep(0.25, length(unique(df.results.b$va
  theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))

p = ggarrange(p3, p4, labels = "AUTO", ncol = 1, nrow = 2)
annotate_figure(p, top = text_grob("Computational faithfulness and model sensitivity: reaction time var
```



Then, we investigated the relationship between the simulated true parameters and the posterior estimates. Although there are individual values diverging from the expected pattern, most parameters were recovered successfully within an uncertainty interval of alpha = 0.05.

Last, we explore the z-score and the posterior contraction of our population-level predictors. The z-score "determines the distance of the posterior mean from the true simulating parameter", while the posterior contraction "estimates how much prior uncertainty is reduced in the posterior estimation" (Schad, Betancourt and Vasisth, 2020). The contraction reveals very narrow posterior standard deviations with slightly lower contraction scores than we would want in the optimal case, however, we judge this as acceptable.

Posterior predictive checks

As the next step, we fit the model, check whether there are divergence or rhat issues, and then check whether the chains have converged.

```r
# fit the final model
set.seed(2684)
m.var = brm(f.var,
            df.var, prior = priors,
            family = lognormal,
            iter = iter, warmup = warm,
            backend = "cmdstanr", threads = threading(8),
            file = "m_var",
            save_pars = save_pars(all = TRUE)
            )
rstan::check_hmc_diagnostics(m.var$fit)
```

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

```
## 0 of 18000 iterations ended with a divergence.
```

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

```
## 0 of 18000 iterations saturated the maximum tree depth of 10.
```

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

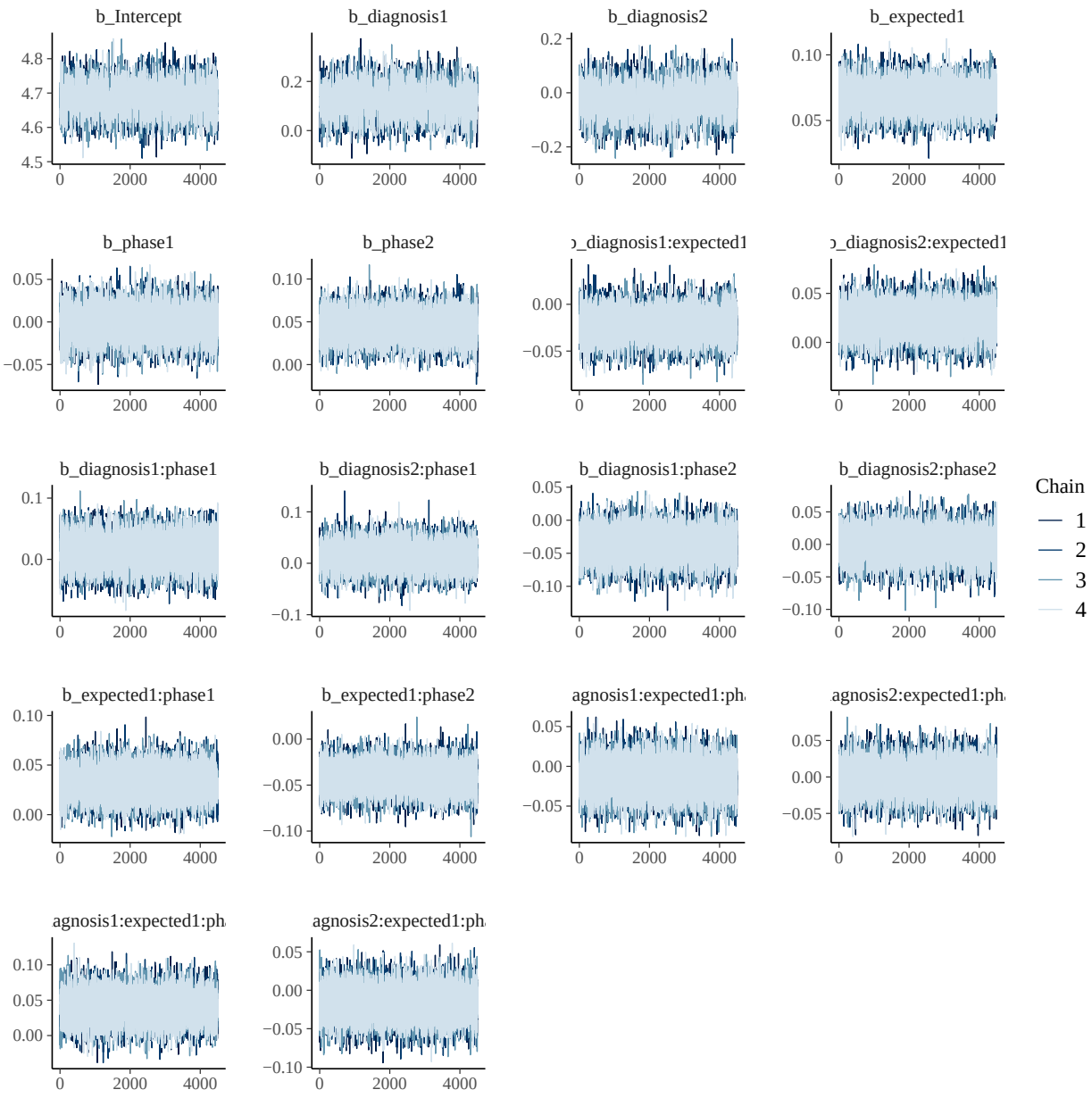
```
## E-BFMI indicated no pathological behavior.
```

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

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

```r
# check the trace plots
post.draws = as_draws_df(m.var)
mcmc_trace(post.draws, regex_pars = "^b_",
           facet_args = list(ncol = 4)) +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
```

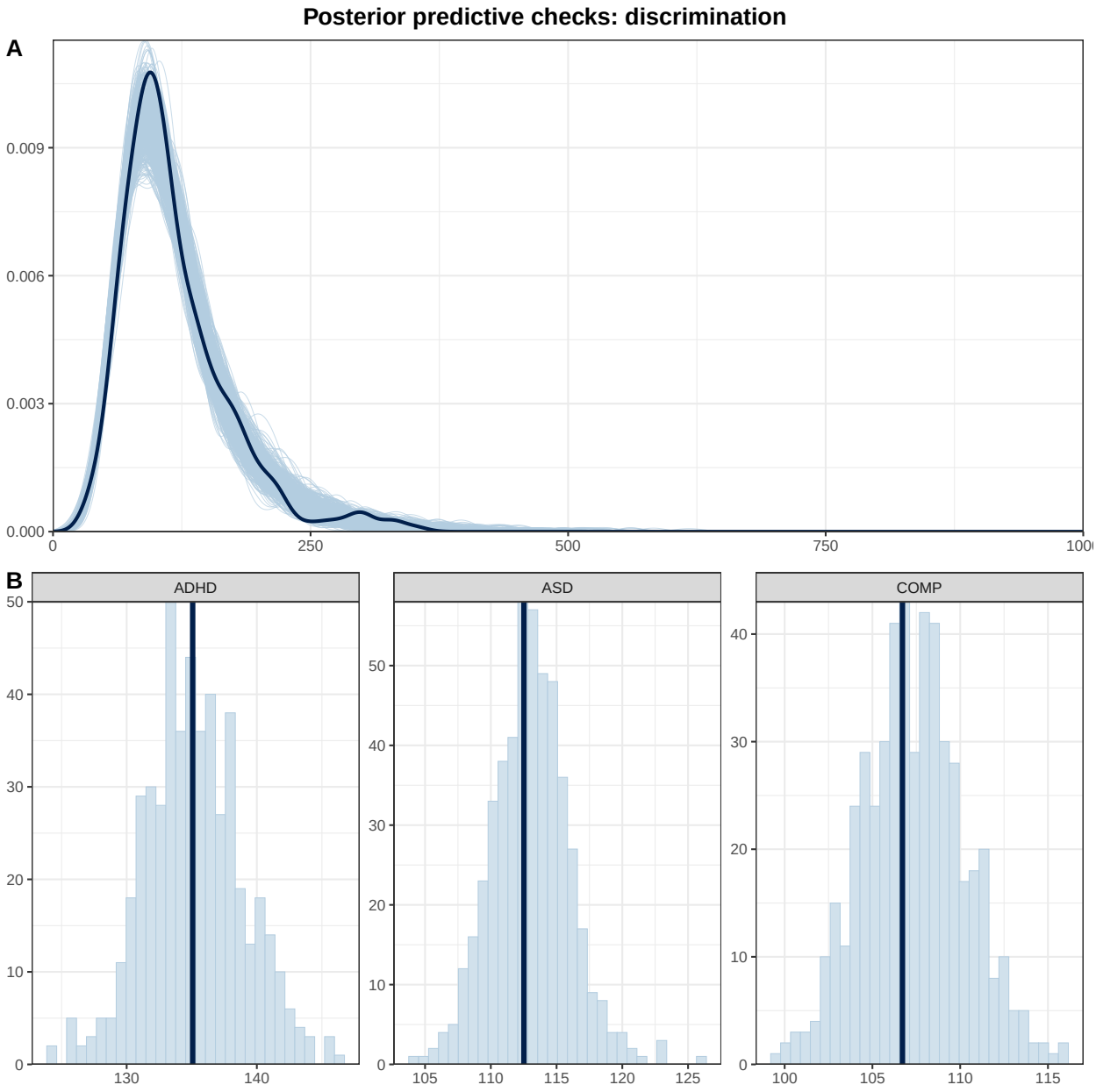
```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```



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

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

# check the fit of the predicted data compared to the real data
p1 = pp_check(m.var, ndraws = nsim) +
  theme_bw() + theme(legend.position = "none") + xlim(0, 1000)

# distributions of means compared to the real values per group
p2 = ppc_stat_grouped(df.var$rt.var, post.pred, df.var$diagnosis) +
  theme_bw() + theme(legend.position = "none")

# distributions of means compared to the real values per group
```

```
p2 = ppc_stat_grouped(df.var$rt.var, post.pred, df.var$diagnosis) +
  theme_bw() + theme(legend.position = "none")

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



The simulated data based on the model fits well with the real data.

Inferences

Now that we are convinced that we can trust our model, we have a look at its estimate and use the hypothesis function to assess our hypotheses and perform explorative tests.

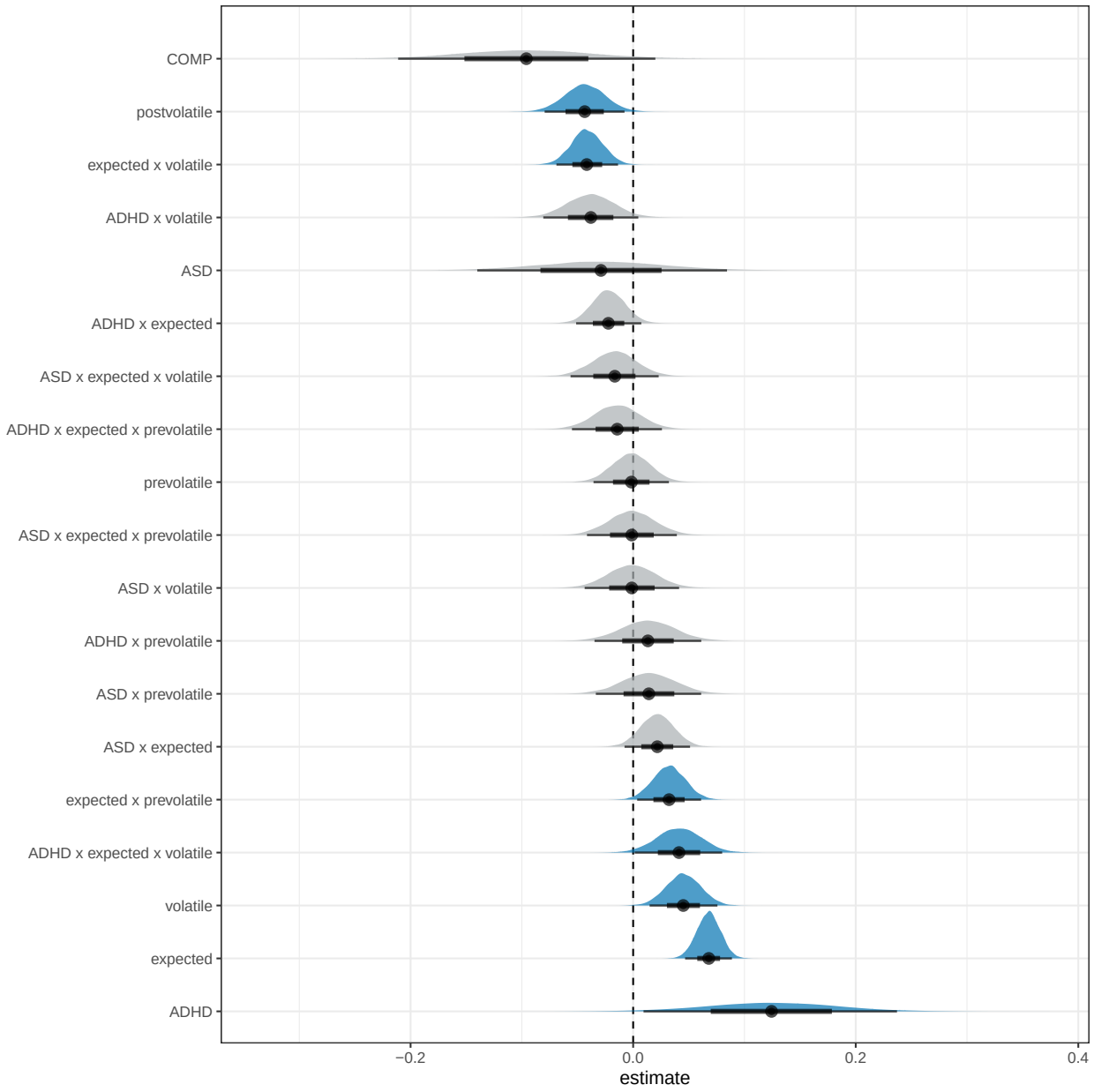
```r
# print a summary
summary(m.var)
```

```
##  Family: lognormal
##   Links: mu = identity; sigma = identity
## Formula: rt.var ~ diagnosis * expected * phase + (expected + phase | subID)
##    Data: df.var (Number of observations: 390)
##   Draws: 4 chains, each with iter = 6000; warmup = 1500; thin = 1;
##          total post-warmup draws = 18000
##
## Multilevel Hyperparameters:
## ~subID (Number of levels: 67)
##                           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)                 0.34      0.03     0.28     0.41 1.00     4951
## sd(expected1)                 0.02      0.01     0.00     0.05 1.00     6023
## sd(phase1)                    0.07      0.02     0.02     0.11 1.00     4423
## sd(phase2)                    0.04      0.02     0.00     0.09 1.00     6087
## cor(Intercept,expected1)     -0.30      0.37    -0.86     0.56 1.00    24331
## cor(Intercept,phase1)        -0.23      0.24    -0.69     0.25 1.00    18469
## cor(expected1,phase1)         0.22      0.41    -0.65     0.87 1.00     3239
## cor(Intercept,phase2)         0.29      0.31    -0.43     0.82 1.00    22100
## cor(expected1,phase2)         0.13      0.43    -0.74     0.84 1.00     7051
## cor(phase1,phase2)            0.13      0.38    -0.64     0.80 1.00    15537
##                           Tail_ESS
## sd(Intercept)                 8569
## sd(expected1)                 8002
## sd(phase1)                    4509
## sd(phase2)                    7695
## cor(Intercept,expected1)     10839
## cor(Intercept,phase1)        11555
## cor(expected1,phase1)         6669
## cor(Intercept,phase2)        11487
## cor(expected1,phase2)        10714
## cor(phase1,phase2)           14432
##
## Regression Coefficients:
##                              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                        4.68      0.04     4.60     4.77 1.00     3297
## diagnosis1                       0.12      0.06     0.01     0.24 1.00     3796
## diagnosis2                      -0.03      0.06    -0.14     0.08 1.00     3667
## expected1                        0.07      0.01     0.05     0.09 1.00    33304
## phase1                          -0.00      0.02    -0.04     0.03 1.00    23372
## phase2                           0.05      0.02     0.01     0.08 1.00    27126
## diagnosis1:expected1            -0.02      0.01    -0.05     0.01 1.00    26478
## diagnosis2:expected1             0.02      0.02    -0.01     0.05 1.00    28285
## diagnosis1:phase1                0.01      0.02    -0.03     0.06 1.00    21374
## diagnosis2:phase1                0.01      0.02    -0.03     0.06 1.00    20594
## diagnosis1:phase2               -0.04      0.02    -0.08     0.00 1.00    24136
## diagnosis2:phase2               -0.00      0.02    -0.04     0.04 1.00    22724
## expected1:phase1                 0.03      0.01     0.00     0.06 1.00    30367
## expected1:phase2                -0.04      0.01    -0.07    -0.01 1.00    31672
## diagnosis1:expected1:phase1     -0.01      0.02    -0.05     0.03 1.00    25954
## diagnosis2:expected1:phase1     -0.00      0.02    -0.04     0.04 1.00    23902
## diagnosis1:expected1:phase2      0.04      0.02     0.00     0.08 1.00    25329
```

```
## diagnosis2:expected1:phase2    -0.02      0.02    -0.06     0.02 1.00    23636
##                              Tail_ESS
## Intercept                        6829
## diagnosis1                       7342
## diagnosis2                       6698
## expected1                       13607
## phase1                          14984
## phase2                          13295
## diagnosis1:expected1            15019
## diagnosis2:expected1            15241
## diagnosis1:phase1               15129
## diagnosis2:phase1               15034
## diagnosis1:phase2               15218
## diagnosis2:phase2               14016
## expected1:phase1                14464
## expected1:phase2                14458
## diagnosis1:expected1:phase1     15378
## diagnosis2:expected1:phase1     15326
## diagnosis1:expected1:phase2     14688
## diagnosis2:expected1:phase2     15223
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.20      0.01     0.18     0.22 1.00     7621    12282
##
## 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 group comparisons
df.m.var = post.draws %>%
  select(starts_with("b_")) %>%
  mutate(
    b_COMP         = - b_diagnosis1 - b_diagnosis2,
    b_postvolatile = - b_phase1 - b_phase2,
    ASD            = b_Intercept + b_diagnosis2,
    ADHD           = b_Intercept + b_diagnosis1,
    COMP           = b_Intercept + b_COMP,
    `ADHD-ASD`     = ADHD - ASD,
    `ADHD-COMP`    = ADHD - COMP,
    `ASD-COMP`     = ASD  - COMP
  )

# plot the posterior distributions
df.m.var %>%
  select(starts_with("b_")) %>%
  pivot_longer(cols = starts_with("b_"), names_to = "coef", values_to = "estimate") %>%
  subset(!startsWith(coef, "b_Int")) %>%
  mutate(
    coef = substr(coef, 3, nchar(coef)),
    coef = str_replace_all(coef, ":", " x "),
    coef = str_replace_all(coef, "diagnosis1", "ADHD"),
    coef = str_replace_all(coef, "diagnosis2", "ASD"),
    coef = str_replace_all(coef, "expected1", "expected"),
```

```
  coef = str_replace_all(coef, "expected2", "unexpected"),
  coef = str_replace_all(coef, "phase1", "prevolatile"),
  coef = str_replace_all(coef, "phase2", "volatile"),
  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(c_dark, c_light)) + theme(legend.position = "none")
```




```
# H1a: COMP < ADHD
h1a = hypothesis(m.var, "0 < 2*diagnosis1 + diagnosis2")
h1a
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis1... < 0    -0.22       0.1    -0.39    -0.05      60.02
##   Post.Prob Star
## 1      0.98    *
## ---
## 'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 95%;
## for two-sided hypotheses, the value tested against lies outside the 95%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

```
## exploration

# expected versus unexpected
e1 = hypothesis(m.var, "0 < expected1", alpha = 0.025)
e1
```

```
## Hypothesis Tests for class b:
##            Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob
## 1 (0)-(expected1) < 0    -0.07      0.01    -0.09    -0.05        Inf         1
##   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.
```

```
# volatile versus prevolatile
e2 = hypothesis(m.var, "0 < phase1 + 2*phase2", alpha = 0.025)
e2
```

```
## Hypothesis Tests for class b:
##                Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(phase1+2*pha... < 0    -0.09      0.03    -0.15    -0.03     946.37
##   Post.Prob Star
## 1         1    *
## ---
## 'CI': 95%-CI for one-sided and 97.5%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 97.5%;
## for two-sided hypotheses, the value tested against lies outside the 97.5%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

```
# prevolatile versus postvolatile
e3 = hypothesis(m.var, "0 < 2*phase1 + phase2", alpha = 0.025)
e3
```

```
## Hypothesis Tests for class b:
##                Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*phase1+pha... < 0    -0.04      0.03     -0.1     0.02       9.59
##   Post.Prob Star
## 1      0.91
## ---
## '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.
```

```
# volatile versus postvolatile
e4 = hypothesis(m.var, "0 > phase1 - phase2", alpha = 0.025)
e4
```

```
## Hypothesis Tests for class b:
##                Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(phase1-phase2) > 0     0.05      0.03    -0.01      0.1       21.7
##   Post.Prob Star
## 1      0.96
## ---
```

```
## '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.
```

```
# ADHD versus ASD
e5 = hypothesis(m.var, "0 < diagnosis1 - diagnosis2", alpha = 0.025)
e5
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(diagnosis1-d... < 0    -0.15       0.1    -0.35     0.04      15.29
##   Post.Prob Star
## 1      0.94
## ---
## 'CI': 95%-CI for one-sided and 97.5%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 97.5%;
## for two-sided hypotheses, the value tested against lies outside the 97.5%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

```
# ASD versus COMP
e6 = hypothesis(m.var, "0 < diagnosis1 + 2*diagnosis2", alpha = 0.025)
e6
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(diagnosis1+2... < 0    -0.07       0.1    -0.26     0.13          3
##   Post.Prob Star
## 1      0.75
## ---
## '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.
```

```
# equivalence of these groups
e1_equ = equivalence_test(df.m.var %>% select(`ADHD-COMP`, `ASD-COMP`, `ADHD-ASD`))
e1_equ
```

```
## # Test for Practical Equivalence
##
##   ROPE: [-0.10 0.10]
##
## Parameter |        H0 | inside ROPE |       95% HDI
## ----------------------------------------------------
## ADHD-COMP | Undecided |     10.02 % |  [0.02, 0.42]
## ASD-COMP  | Undecided |     61.16 % | [-0.13, 0.26]
## ADHD-ASD  | Undecided |     28.15 % | [-0.04, 0.35]
```

```
## extract predicted differences in ms instead of log data
df.new = df.var %>%
  select(diagnosis, phase, expected) %>%
  distinct() %>%
  mutate(
    condition = paste(diagnosis, phase, expected, sep = "_")
  )
df.ms = as.data.frame(
```

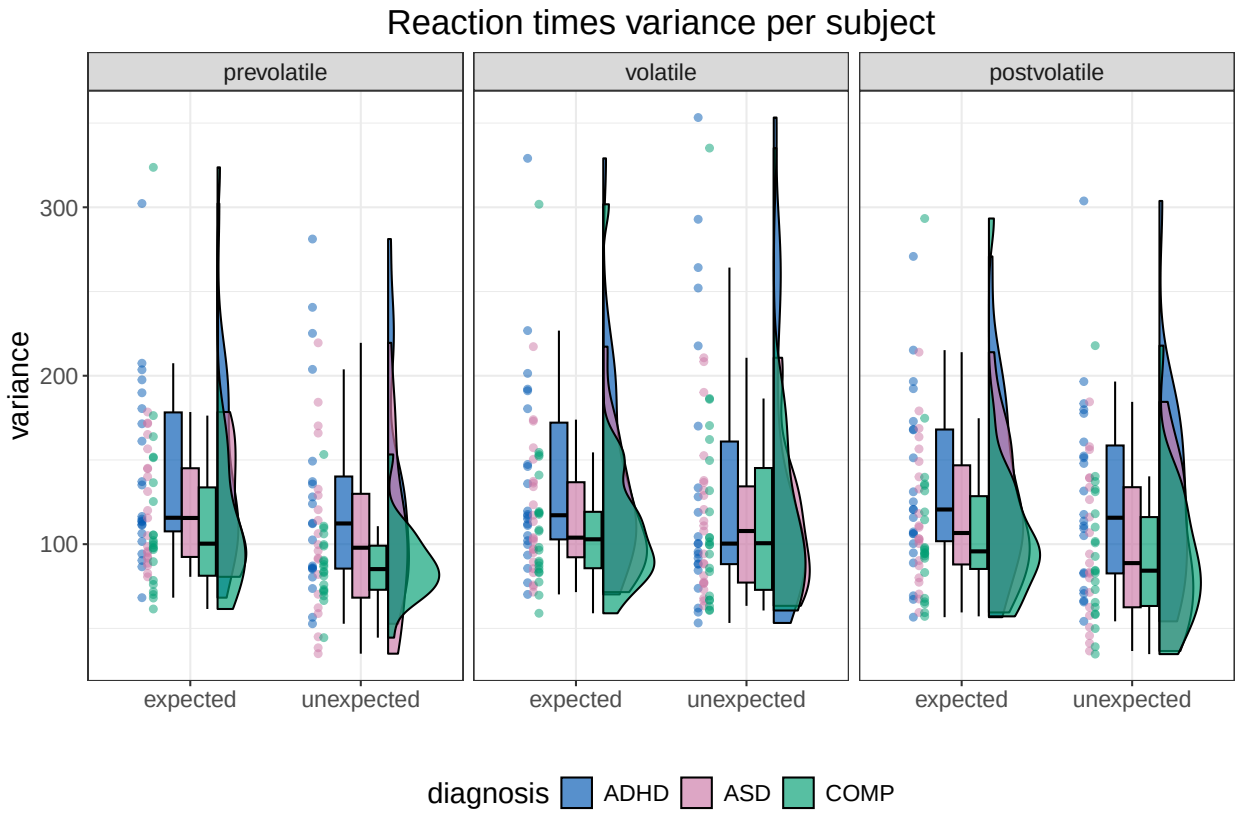
```
  fitted(m.var, summary = F,
               newdata = df.new %>% select(diagnosis, phase, expected),
               re_formula = NA))
colnames(df.ms) = df.new$condition

# calculate our difference columns
df.ms = df.ms %>%
  mutate(
    COMP          = rowMeans(across(matches("COMP_.*"))),
    ADHD          = rowMeans(across(matches("ADHD_.*"))),
    ASD           = rowMeans(across(matches("ASD_.*"))),
    h1a_COMPvADHD = ADHD - COMP,
    exp           = rowMeans(across(matches(".*_expected"))),
    unexp         = rowMeans(across(matches(".*_unexpected"))),
    e1_expvunexp  = exp - unexp,
    volatile      = rowMeans(across(matches(".*_volatile_.*"))),
    prevolatile   = rowMeans(across(matches(".*_prevolatile_.*"))),
    postvolatile  = rowMeans(across(matches(".*_postvolatile_.*"))),
    e2_volvpre    = volatile - prevolatile,
    e3_postvpre   = postvolatile - prevolatile,
    e4_volvpost   = volatile - postvolatile,
    e5_ADHDvASD   = ADHD - ASD,
    e6_COMPvASD   = COMP - ASD
  )

lapply(df.ms %>% select(starts_with("e") | starts_with("h")), ci)
```

```
## $exp
## 95% ETI: [109.10, 129.42]
##
## $e1_expvunexp
## 95% ETI: [9.81, 19.35]
##
## $e2_volvpre
## 95% ETI: [-1.71, 10.65]
##
## $e3_postvpre
## 95% ETI: [-11.49, 2.24]
##
## $e4_volvpost
## 95% ETI: [2.71, 15.59]
##
## $e5_ADHDvASD
## 95% ETI: [-5.42, 40.40]
##
## $e6_COMPvASD
## 95% ETI: [-27.83, 13.82]
##
## $h1a_COMPvADHD
## 95% ETI: [1.70, 47.38]
```

[!MISSING]

Plots

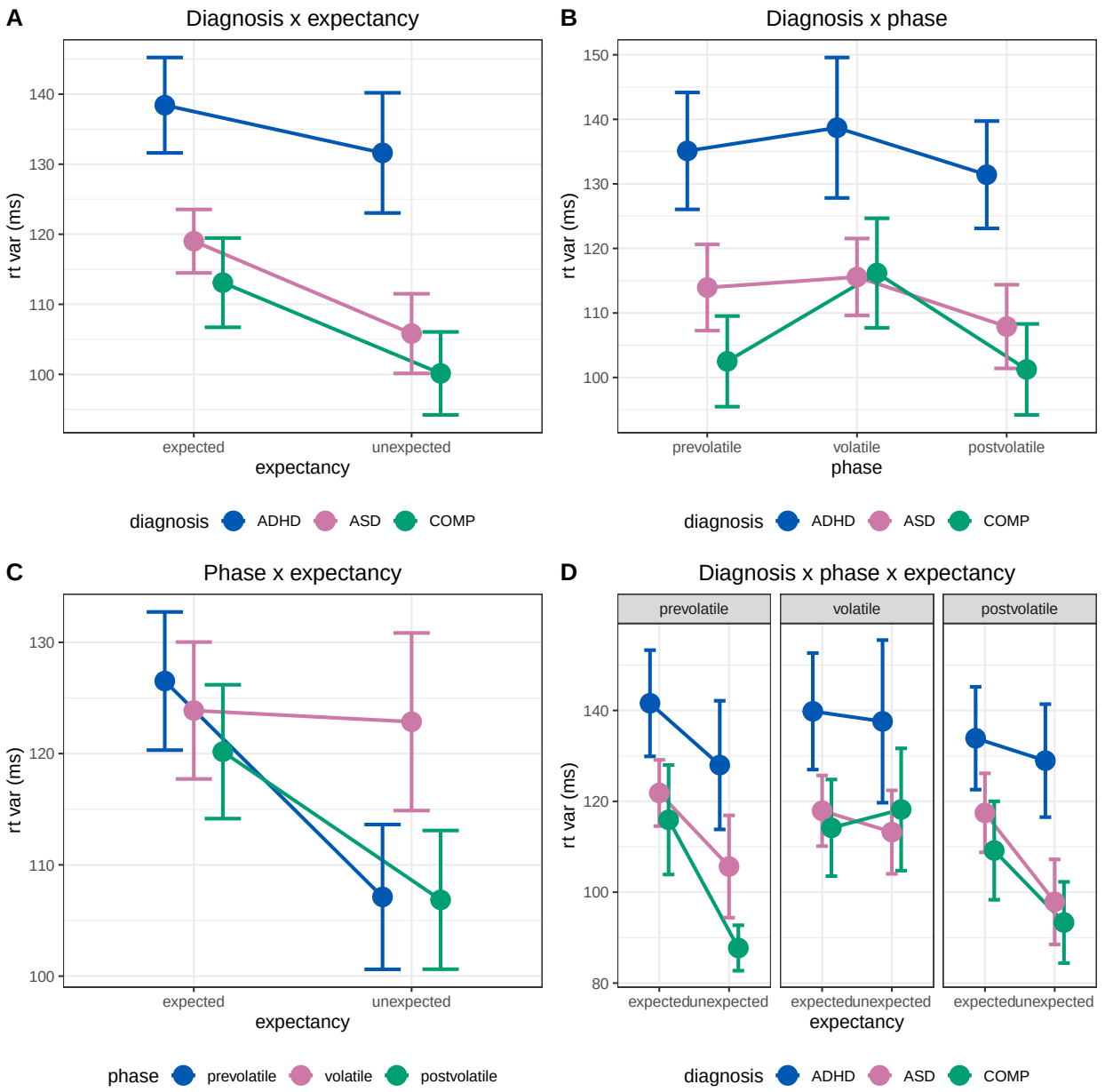
```
# rain cloud plot
a = 0.66
df.var %>%
  ggplot(aes(expected, rt.var, fill = diagnosis, colour = diagnosis)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show.legend = FALSE, alpha = a),
violin.args = list(color = "black", outlier.shape = NA, alpha = a),
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.col) +
  scale_color_manual(values = custom.col) +
  #ylim(0, 1) +
  labs(title = "Reaction times variance per subject", x = "", y = "variance") +
  facet_wrap(. ~ phase) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5), legend.direction = "horizon
```



Now, we use line plots to visualise out the different main effects and interaction effects in our data.

```r
# two-way interactions
p1 = df.var %>%
  group_by(diagnosis, expected) %>%
  summarise(
    var.mn = mean(rt.var),
    var.se = sd(rt.var) / sqrt(n())
  ) %>%
  ggplot(aes(y = var.mn, x = expected, group = diagnosis, colour = diagnosis)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = var.mn - var.se,
                    ymax = var.mn + var.se), linewidth = 1, width = 0.5,
                position = position_dodge(0.4)) +
  geom_point(position = position_dodge(0.4), size = 5) +
  labs (x = "expectancy", y = "rt var (ms)") +
  ggtitle("Diagnosis x expectancy") +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))
p2 = df.var %>%
  group_by(diagnosis, phase) %>%
  summarise(
    var.mn = mean(rt.var),
    var.se = sd(rt.var) / sqrt(n())
  ) %>%
  ggplot(aes(y = var.mn, x = phase, group = diagnosis, colour = diagnosis)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = var.mn - var.se,
                    ymax = var.mn + var.se), linewidth = 1, width = 0.5,
                position = position_dodge(0.4)) +
  geom_point(position = position_dodge(0.4), size = 5) +
  labs (x = "phase", y = "rt var (ms)") +
  ggtitle("Diagnosis x phase") +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))
p3 = df.var %>%
  group_by(phase, expected) %>%
  summarise(
    var.mn = mean(rt.var),
    var.se = sd(rt.var) / sqrt(n())
  ) %>%
  ggplot(aes(y = var.mn, x = expected, group = phase, colour = phase)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = var.mn - var.se,
                    ymax = var.mn + var.se), linewidth = 1, width = 0.5,
                position = position_dodge(0.4)) +
  geom_point(position = position_dodge(0.4), size = 5) +
  labs (x = "expectancy", y = "rt var (ms)") +
  ggtitle("Phase x expectancy") +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
```

```
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))

# three-way interaction
p4 =  df.var %>%
  group_by(diagnosis, expected, phase) %>%
  summarise(
    var.mn = mean(rt.var),
    var.se = sd(rt.var) / sqrt(n())
  ) %>%
  ggplot(aes(y = var.mn, x = expected, group = diagnosis, colour = diagnosis)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = var.mn - var.se,
                    ymax = var.mn + var.se), linewidth = 1, width = 0.5,
                position = position_dodge(0.4)) +
  geom_point(position = position_dodge(0.4), size = 5) +
  labs (x = "expectancy", y = "rt var (ms)") +
  ggtitle("Diagnosis x phase x expectancy") +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  facet_wrap(. ~ phase) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))

ggarrange(p1, p2, p3, p4, ncol = 2, nrow = 2, labels = "AUTO")
```



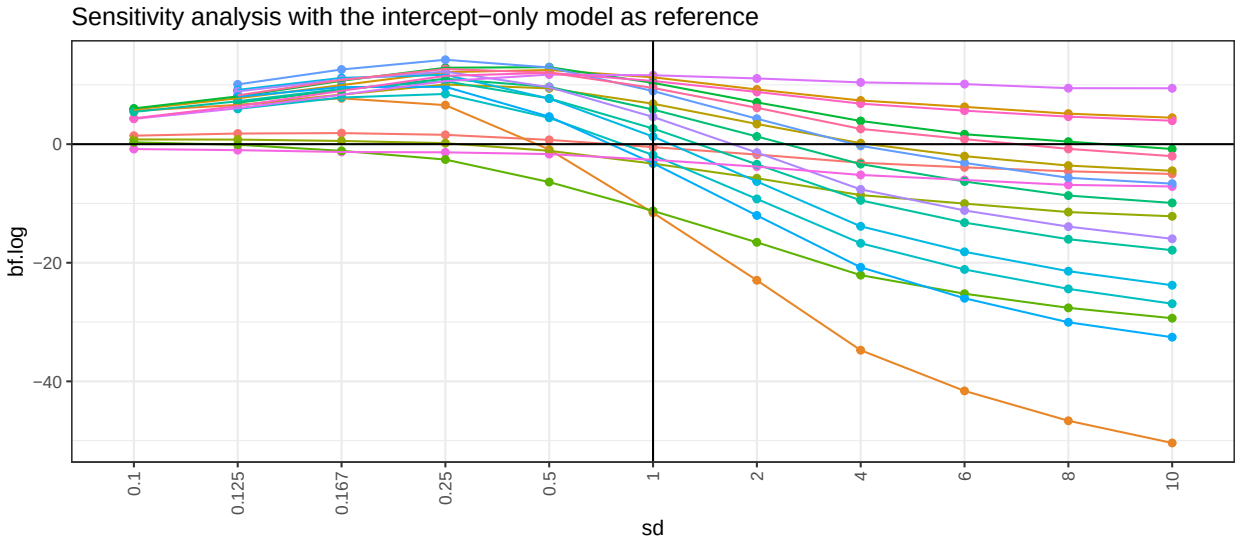
Bayes factor analysis

To complement our hypothesis testing using brms::hypothesis(), we perform a Bayes Factor (BF) analysis. The BF is the ratio of the marginal likelihoods of the data given two models. We will compare models containing different combinations of population-level effects to the model only containing the intercept on the population-level and all group-level effects. The BF depends on the priors that were used, because it indicates a change in our belief after seeing the data. Therefore, we perform a sensitivity analysis comparing the BF based on our chosen priors with narrower and wider priors.

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

# rerun the model with more iterations for bridgesampling
```

```r
set.seed(2684)
m.var.bf = brm(f.var,
               df.var, prior = priors,
               family = lognormal,
               iter = iter, warmup = warm,
               backend = "cmdstanr", threads = threading(8),
               file = "m_var_bf",
               save_pars = save_pars(all = TRUE)
               )

# describe priors for the sensitivity analysis
pr.descriptions = c("chosen",
  "sdx2",    "sdx4",   "sdx6",     "sdx8",     "sdx10",
  "sdx0.5", "sdx0.25", "sdx0.167", "sdx0.125", "sdx0.1"
  )

# check which have been run already
if (file.exists(file.path(sense_dir, sprintf("df_%s_bf.csv", main.code)))) {
  pr.done = read_csv(
    file.path(sense_dir, sprintf("df_%s_bf.csv", main.code)),
    show_col_types = F) %>%
    select(priors) %>% distinct()
  pr.descriptions = pr.descriptions[!(pr.descriptions %in% pr.done$priors)]
}

# loop through the priors that have not been used before
for (pr.desc in pr.descriptions) {
  tryCatch({
    # use function to compute BF with the described priors
    bf_sens_3int(m.var.bf, "diagnosis", "phase", "expected",
                 pr.desc,
                 main.code, # prefix for all models and MLL
                 file.path("/home/emba/Insync/10planki@gmail.com/Google Drive/NEVIA/logfiles", "log_PAL
                 sense_dir # where to save the models and MLL
    )
  },
  error = function(err) {
    message(sprintf("Error for %s: %s", pr.desc, err))
  }
  )
}

df.var.bf = read_csv(file.path(sense_dir,
                               sprintf("df_%s_bf.csv", main.code)),
                     show_col_types = F)

# check the sensitivity analysis result per model
df.var.bf %>%
  filter(`population-level` != "1") %>%
  mutate(
    sd = as.factor(case_when(
      priors == "chosen" ~ "1",
      substr(priors, 1, 3) == "sdx" ~ gsub("sdx", "", priors),
```

```
      T ~ priors)
    ),
    order = case_when(
      priors == "chosen" ~ 1,
      substr(priors, 1, 3) == "sdx" ~ as.numeric(gsub("sdx", "", priors)),
      T ~ 999),
    sd = fct_reorder(sd, order)
  ) %>%
  ggplot(aes(y = bf.log,
             x = sd,
             group = `population-level`,
             colour = `population-level`)) +
  geom_point() +
  geom_line() +
  geom_vline(xintercept = "1") +
  geom_hline(yintercept = 0) +
  ggtitle("Sensitivity analysis with the intercept-only model as reference") +
  #scale_colour_manual(values = custom.col) +
  theme_bw() +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```




```
# create a data frame with the comparisons
kable(head(df.var.bf %>% filter(priors == "chosen") %>% select(-priors) %>%
  filter(`population-level` != "1") %>% arrange(desc(bf.log))), digits = 3)
```

population-level	bf.log
expected	11.624
diagnosis + expected	11.276
phase + expected	10.656
diagnosis + phase + expected	10.336
phase + expected + phase:expected	9.483
diagnosis + phase + expected + phase:expected	8.955

S1.3 Reaction times in correct trials

In the preregistration, we noted the following population-level effects for the model of the reaction time variances: group, expectancy, phase and difficulty; as well as the group-level predictores subject and trials. However, the posterior fit for the full model was suboptimal, therefore, we aggregated the reaction times for correct trials using the median (see above in the preparation of the data).

```
# figure out slopes for subjects
kable(head(df.pal %>% count(subID, expected)))
```

subID	expected	n
1	expected	9
1	unexpected	9
2	expected	9
2	unexpected	9
3	expected	9
3	unexpected	9

```
kable(head(df.pal %>% count(subID, phase)))
```

subID	phase	n
1	prevolatile	6
1	volatile	6
1	postvolatile	6
2	prevolatile	6
2	volatile	6
2	postvolatile	6

```
kable(head(df.pal %>% count(subID, difficulty)))
```

subID	difficulty	n
1	easy	6
1	medium	6
1	difficult	6
2	easy	6
2	medium	6
2	difficult	6

```
kable(head(df.pal %>% count(subID, expected, phase)))
```

subID	expected	phase	n
1	expected	prevolatile	3
1	expected	volatile	3
1	expected	postvolatile	3
1	unexpected	prevolatile	3
1	unexpected	volatile	3
1	unexpected	postvolatile	3

```
kable(head(df.pal %>% count(subID, expected, difficulty)))
```

subID	expected	difficulty	n
1	expected	easy	3
1	expected	medium	3
1	expected	difficult	3
1	unexpected	easy	3
1	unexpected	medium	3
1	unexpected	difficult	3

```
kable(head(df.pal %>% count(subID, phase, difficulty)))
```

subID	phase	difficulty	n
1	prevolatile	easy	2
1	prevolatile	medium	2
1	prevolatile	difficult	2
1	volatile	easy	2
1	volatile	medium	2
1	volatile	difficult	2

```
kable(head(df.pal %>% count(subID, expected, phase, difficulty)))
```

subID	expected	phase	difficulty	n
1	expected	prevolatile	easy	1
1	expected	prevolatile	medium	1
1	expected	prevolatile	difficult	1
1	expected	volatile	easy	1
1	expected	volatile	medium	1
1	expected	volatile	difficult	1

```
code = "PAL-rt"

# set the formula
f.pal = brms::bf(rt.cor ~ diagnosis * expected * phase * difficulty +
                   (expected + phase + difficulty +
                      expected:phase + difficulty:phase + expected:difficulty | subID))

# set informed priors based on previous results
priors = c(
  # informative priors based Lawson et al. and Schad, Betancourt & Vasishth (2019)
  prior(normal(6.0,   0.3),   class = Intercept),
  prior(normal(0.0,   0.5),   class = sigma),
  prior(normal(0,     0.1),   class = sd),
  prior(lkj(2),               class = cor),
  prior(normal(100, 100.0),   class = ndt),
  # ASD slower overall (Lawson et al., 2017)
  prior(normal(0.02,  0.04),  class = b, coef = diagnosis2),
  # faster for expected trials (Lawson et al., 2017)
  prior(normal(-0.02, 0.04),  class = b, coef = expected1), # expected
```

```
  # faster on easy trials (Lawson et al., 2017)
  prior(normal(-0.02, 0.04),  class = b, coef = difficulty1), # easy
  # larger effect of phases in ASD (Shi et al., 2022)
  prior(normal(0.02,  0.04),  class = b, coef = diagnosis2:phase2),
  prior(normal(0.02,  0.04),  class = b, coef = diagnosis2:phase1),
  # smaller effect of expected in ASD (Lawson et al., 2017)
  prior(normal(0.02,  0.04),  class = b, coef = diagnosis2:expected1),
  # all the other interactions
  prior(normal(0.00,  0.04),  class = b)
)
```

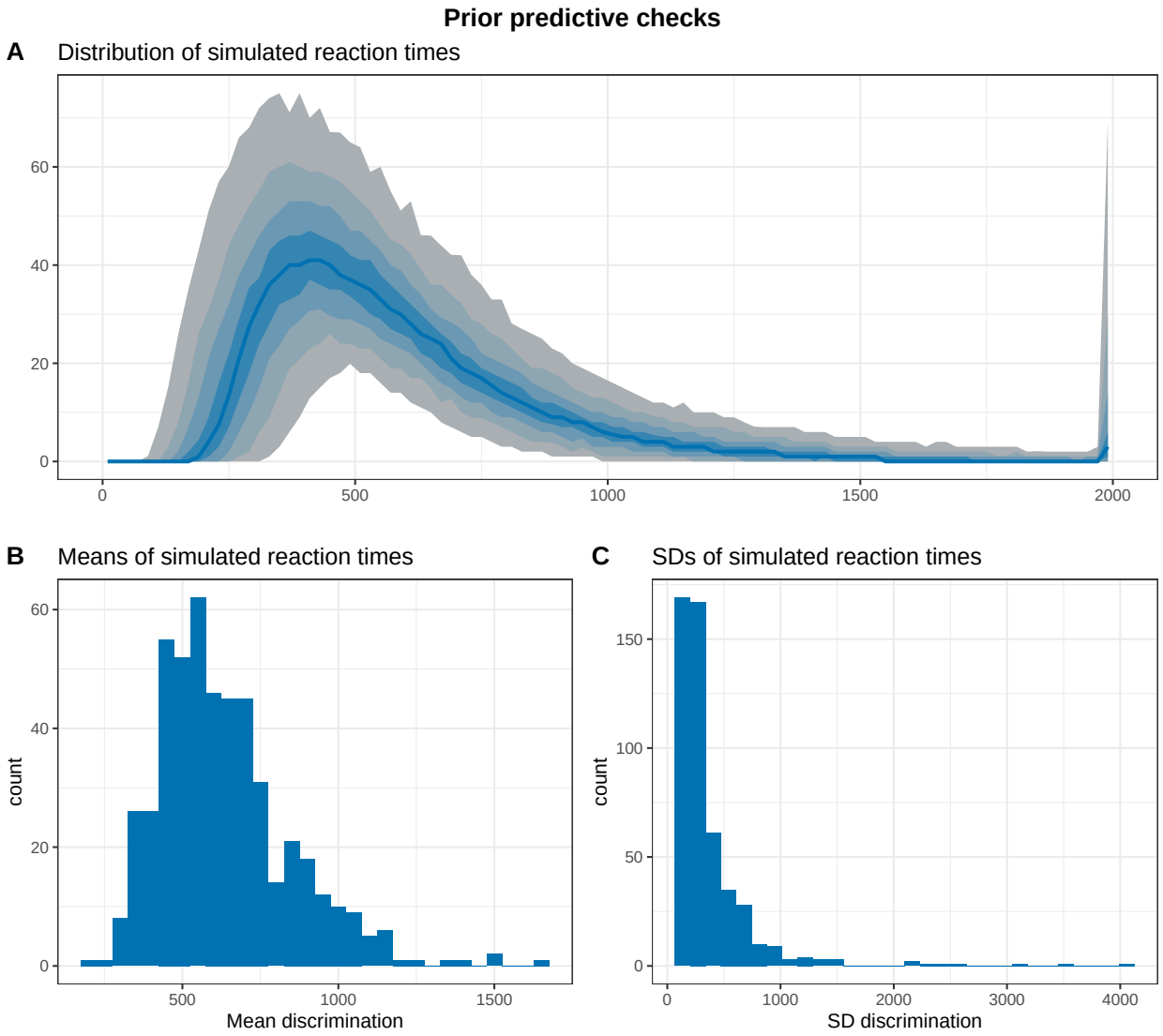
Simulation-based calibration

```
if (!(file.exists(file.path(cache_dir, ""))) | !(file.exists(file.path(cache_dir)))) {
  # simulate some data based on the priors
  gen = SBC_generator_brms(f.pal, data = df.pal, prior = priors,
                           thin = 50, warmup = 10000, refresh = 2000,
                           generate_lp = TRUE, family = shifted_lognormal, init = 0.1)
  if (!file.exists(file.path(cache_dir, paste0("dat_", code, ".rds")))){
    dat = generate_datasets(gen, nsim)
    saveRDS(dat, file = file.path(cache_dir, paste0("dat_", code, ".rds")))
  } else {
    dat = readRDS(file = file.path(cache_dir, paste0("dat_", code, ".rds")))
  }

  # perform the SBC
  bck = SBC_backend_brms_from_generator(gen, chains = 4, thin = 1,
                                        warmup = warm, iter = iter,
                                        init = 0.1)
  # only do it for 250 simulations since it takes so long
  # > this was done with the helper script, seeds are documented there!
  res = compute_SBC(SBC_datasets(dat$variables[1:250,],
                                 dat$generated[1:250]),
                    bck,
                    cache_mode = "results",
                    cache_location = file.path(cache_dir, sprintf("res_%s", code, i)))
  write_csv(res$stats, file.path(cache_dir, sprintf("df_res_%s.rds", code)))
  write_csv(res$backend_diagnostics, file.path(cache_dir, sprintf("df_div_%s.rds", code)))
} else {
  # 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)))
}
```

Again, we start by investigating the rhats and the number of divergent samples. This shows that 2 of 500 simulations had at least one parameter that had an rhat of at least 1.05, and 2 models had divergent samples. This suggests that this model performs well and we can continue with our checks by plotting the simulated values to perform prior predictive checks.

```
if (!(file.exists(file.path(cache_dir, sprintf("dvfakemat_%s", code))))) {
  # create a matrix out of generated data
  dvname = gsub(" ", "", gsub("[\\|~].*", "", f.pal)[1])
  dvfakemat = matrix(NA, nrow(dat[['generated']][[1]]), length(dat[['generated']]))
  for (i in 1:length(dat[['generated']])) {
```

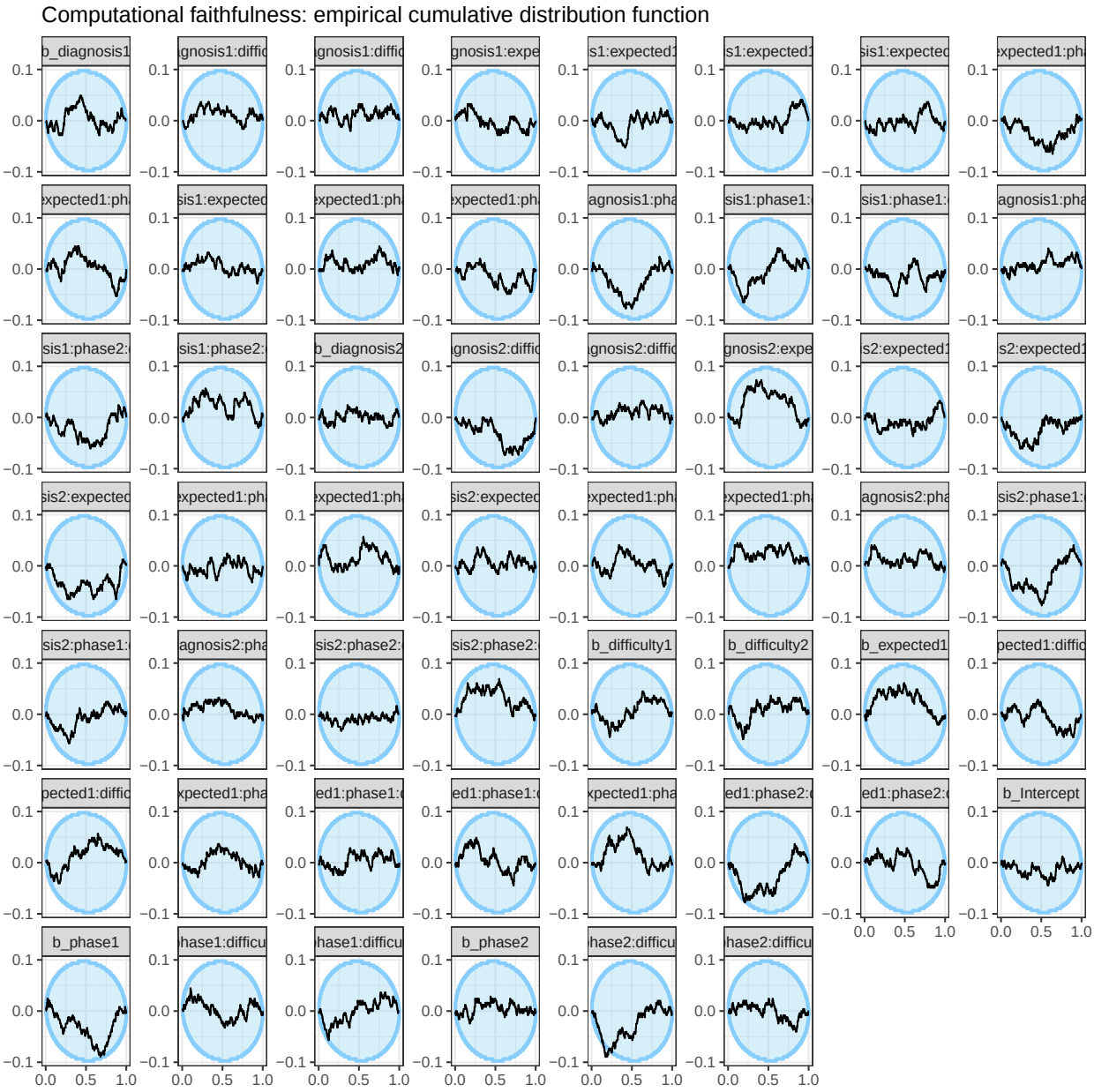
```
    dvfakemat[,i] = dat[['generated']][[i]][[dvname]]
  }
  saveRDS(dvfakemat, file.path(cache_dir, sprintf("dvfakemat_%s", code)))
} else {
  dvfakemat = readRDS(file.path(cache_dir, sprintf("dvfakemat_%s", code)))
}

# compute one histogram per simulated data-set
dvfakematH = dvfakemat
dvfakematH[dvfakematH > 2000] = 2000
breaks = seq(0, max(dvfakematH, na.rm=T), length.out = 101)
binwidth = breaks[2] - breaks[1]
histmat = matrix(NA, ncol = dim(dvfakemat)[2], nrow = length(breaks)-1)
for (i in 1:dim(dvfakemat)[2]) {
  histmat[,i] = hist(dvfakematH[,i], breaks = breaks, plot = F)$counts
}
# for each bin, compute quantiles across histograms
probs = seq(0.1, 0.9, 0.1)
quantmat = as.data.frame(matrix(NA, nrow=dim(histmat)[1], ncol = length(probs)))
names(quantmat) = paste0("p", probs)
for (i in 1:dim(histmat)[1]) {
  quantmat[i,] = quantile(histmat[i,], p = probs)
}
quantmat$x = breaks[2:length(breaks)] - binwidth/2 # add bin mean
p1 = ggplot(data = quantmat, aes(x = x)) +
  geom_ribbon(aes(ymax = p0.9, ymin = p0.1), fill = c_light) +
  geom_ribbon(aes(ymax = p0.8, ymin = p0.2), fill = c_light_highlight) +
  geom_ribbon(aes(ymax = p0.7, ymin = p0.3), fill = c_mid) +
  geom_ribbon(aes(ymax = p0.6, ymin = p0.4), fill = c_mid_highlight) +
  geom_line(aes(y = p0.5), colour = c_dark, linewidth = 1) +
  labs(title = "Distribution of simulated reaction times", y = "", x = "") +
  theme_bw()

tmpM = apply(dvfakemat, 2, mean) # mean
tmpSD = apply(dvfakemat, 2, sd)
p2 = ggplot() +
  stat_bin(aes(x = tmpM), fill = c_dark)  +
  labs(x = "Mean discrimination", title = "Means of simulated reaction times") +
  theme_bw()
p3 = ggplot() +
  stat_bin(aes(x = tmpSD), fill = c_dark) +
  labs(x = "SD discrimination", title = "SDs of simulated reaction times") +
  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))
```



Subfigure A shows the distribution of the simulated data with bluer bands being more likely than greyer bands. Subfigure A shows a distribution that fits our expectations about reaction times in a simple decision task. The distribution of the means and standard deviations in the simulated datasets also look good. We go ahead with these priors and check the results of the SBC. We only plot the results from the models that had no divergence issues.

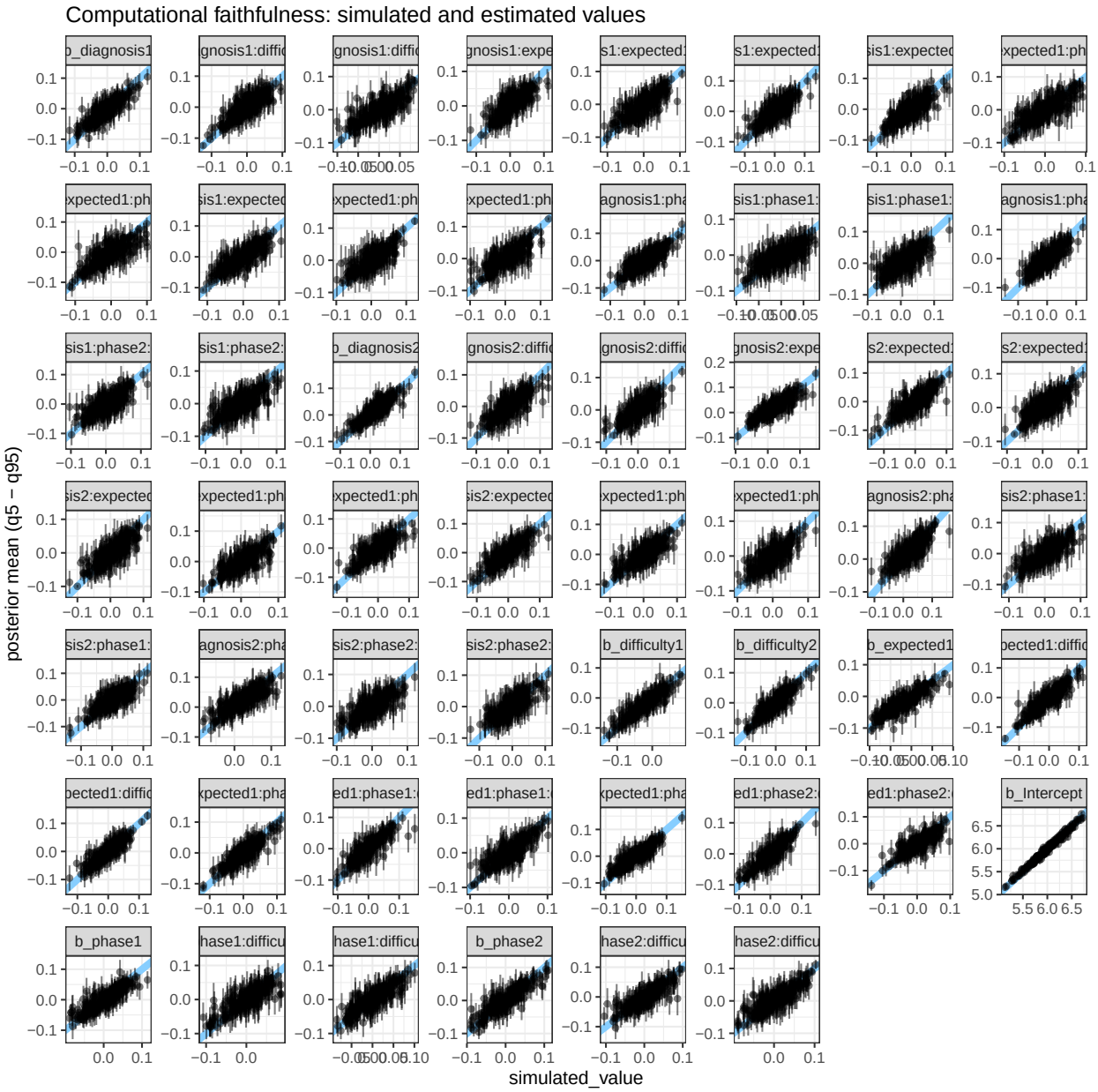
```
# get simulation numbers with issues
rank = max(df.results$max_rank)
check = merge(df.results %>%
    group_by(sim_id) %>%
      summarise(
        rhat = max(rhat, na.rm = T),
        mean_rank = mean(max_rank)
        ) %>%
    filter(rhat >= 1.05 | mean_rank != rank),
  df.backend %>% filter(n_divergent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

# plot SBC with functions from the SBC package focusing on population-level parameters
df.results.b = df.results %>%
```

```
  filter(substr(variable, 1, 2) == "b_") %>%
  filter(!(sim_id %in% check$sim_id))
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)) +
  ggtitle("Computational faithfulness: empirical cumulative distribution function")
```

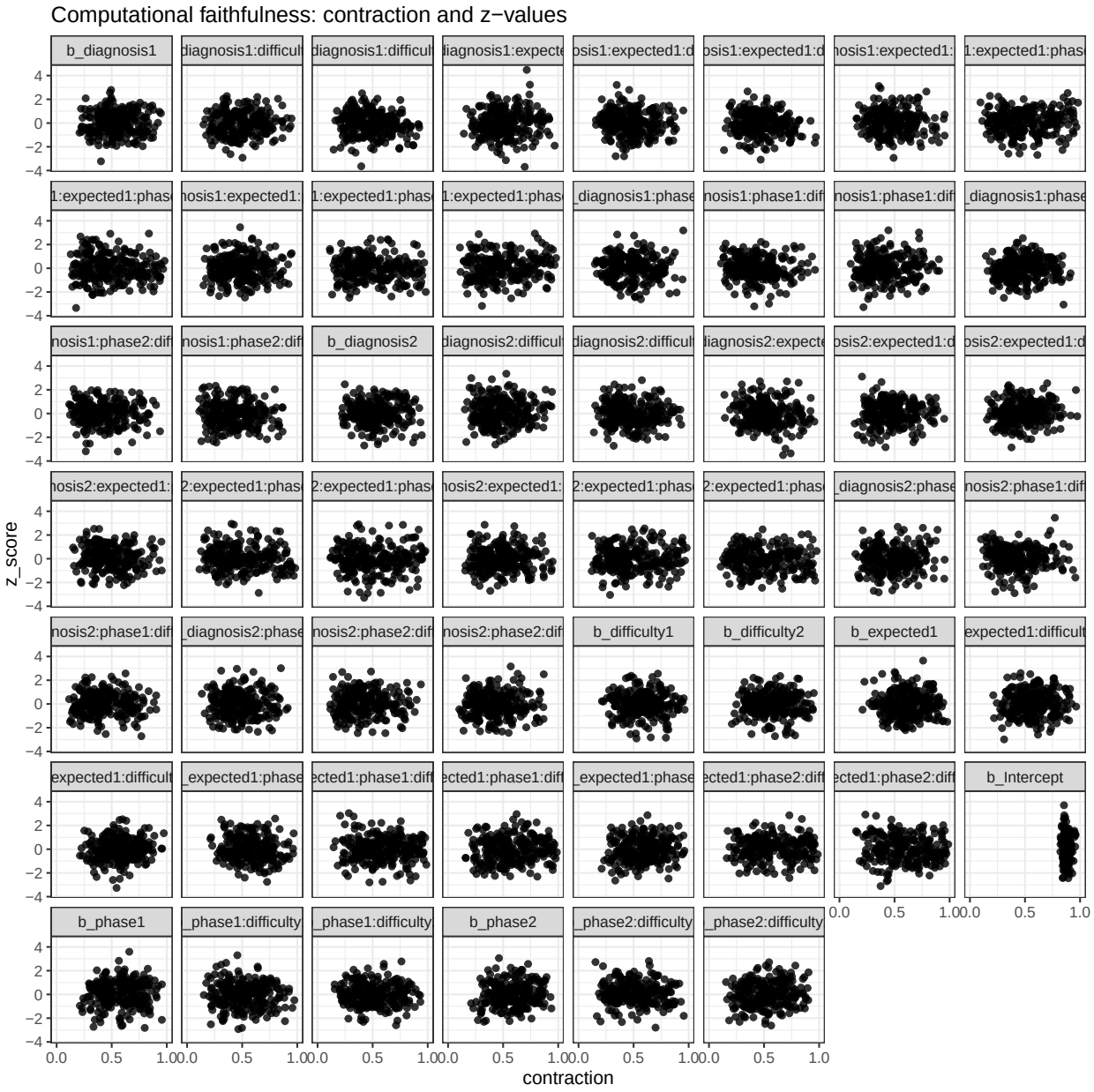



```
plot_rank_hist(df.results.b, bins = 20) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3)) +
  ggtitle("Computational faithfulness: rank histograms")
```


```
plot_sim_estimated(df.results.b, alpha = 0.5) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3)) +
  ggtitle("Computational faithfulness: simulated and estimated values")
```




```
plot_contraction(df.results.b,
                 prior_sd = setNames(
                   c(as.numeric(
                     gsub(".*, (.+)\\).*", "\\1",
                          priors[priors$class == "Intercept",]$prior)),
                   rep(0.04, times = (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 = 5)) +
  ggtitle("Computational faithfulness: contraction and z-values")
```



Next, we check the ranks of the parameters. If the model is unbiased, these should be uniformly distributed (Schad, Betancourt and Vasishth, 2020). The sample empirical cumulative distribution function (ECDF) lies within the theoretical distribution (95%) and the rank histogram also shows ranks within the 95% expected range, although there are some small deviations. We judge this to be acceptable.

Then, we investigated the relationship between the simulated true parameters and the posterior estimates. Although there are individual values diverging from the expected pattern, most parameters were recovered successfully within an uncertainty interval of alpha = 0.05.

Last, we explore the z-score and the posterior contraction of our population-level predictors. The z-score "determines the distance of the posterior mean from the true simulating parameter", while the posterior contraction "estimates how much prior uncertainty is reduced in the posterior estimation" (Schad, Betancourt and Vasisth, 2020). All of this looks good for this model.

Posterior predictive checks

As the next step, we fit the model, check whether there are divergence or rhat issues, and then check whether the chains have converged.

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

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

```
## 0 of 18000 iterations ended with a divergence.
```

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

```
## 0 of 18000 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.pal) >= 1.01, na.rm = T)
```

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

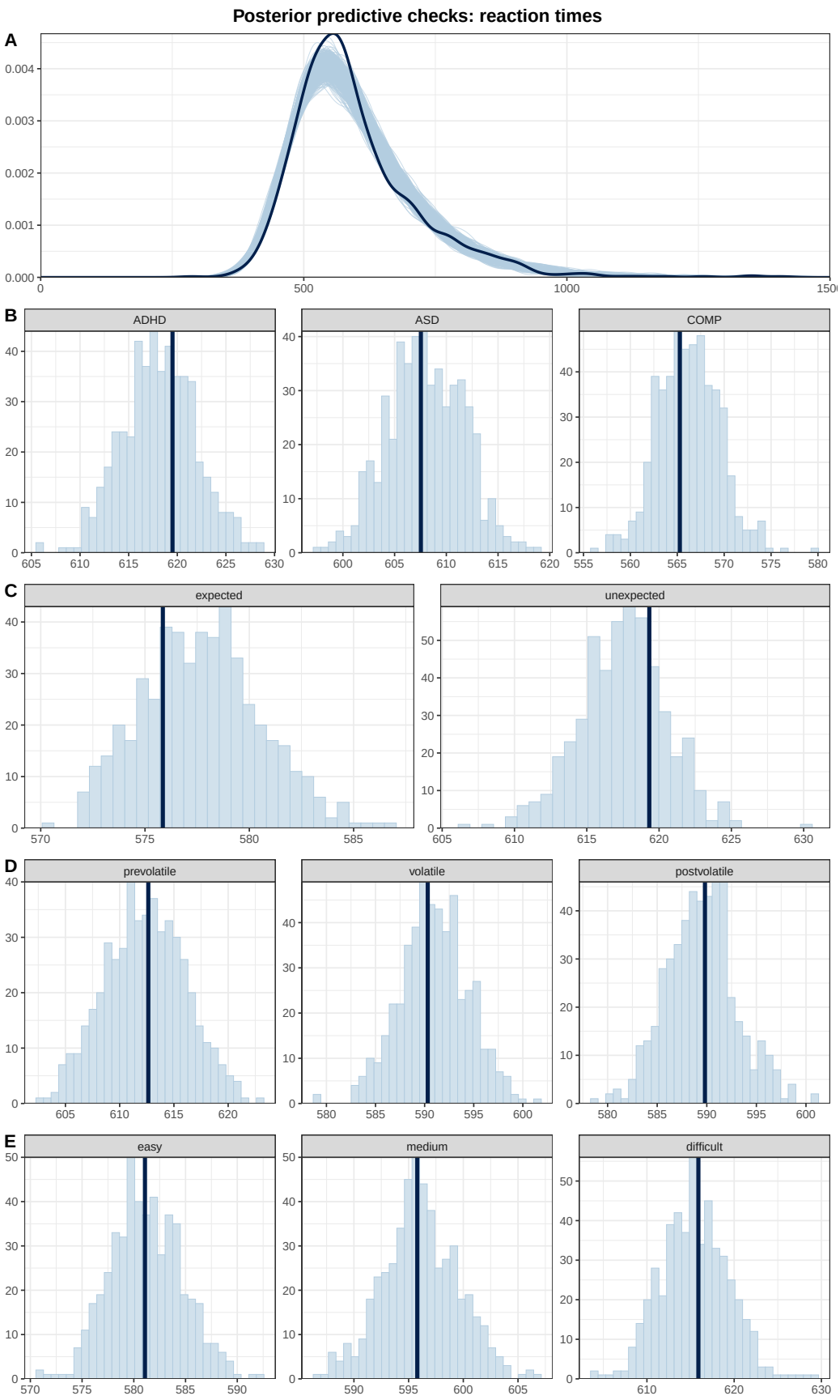
```
# check the trace plots
post.draws = as_draws_df(m.pal)
mcmc_trace(post.draws, regex_pars = "^b_",
           facet_args = list(ncol = 6)) +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
```

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

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

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

# check the fit of the predicted data compared to the real data
p1 = pp_check(m.pal, ndraws = nsim) +
  theme_bw() + theme(legend.position = "none") + xlim(0, 1500)

# distributions of means compared to the real values per group or conditions
p2 = ppc_stat_grouped(df.pal$rt.cor,
                      post.pred,
                      df.pal$diagnosis) +
  theme_bw() + theme(legend.position = "none")
```

```
p3 = ppc_stat_grouped(df.pal$rt.cor,
                      post.pred,
                      df.pal$expected) +
  theme_bw() + theme(legend.position = "none")
p4 = ppc_stat_grouped(df.pal$rt.cor,
                      post.pred,
                      df.pal$phase) +
  theme_bw() + theme(legend.position = "none")
p5 = ppc_stat_grouped(df.pal$rt.cor,
                      post.pred,
                      df.pal$difficulty) +
  theme_bw() + theme(legend.position = "none")

p = ggarrange(p1, p2, p3, p4, p5,
          nrow = 5, ncol = 1, labels = "AUTO")
annotate_figure(p, top = text_grob("Posterior predictive checks: reaction times",
                                   face = "bold", size = 14))
```


Posterior predictive checks: reaction times
A
B
ADHD
ASD
COMP
C
expected
unexpected
D
prevolatile
volatile
postvolatile
E
easy
medium
difficult

The simulated data based on the model fits well with the real data, although there are slight deviations specifically for the predictor expectedness. However, it is much better than for the full model, so we judge this to be acceptable.

Inferences

Now that we are convinced that we can trust our model, we have a look at its estimate and use the hypothesis function to assess our hypotheses and perform explorative tests.

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

```
##  Family: shifted_lognormal
##   Links: mu = identity; sigma = identity; ndt = identity
## Formula: rt.cor ~ diagnosis * expected * phase * difficulty + (expected + phase + difficulty + expec
##    Data: df.pal (Number of observations: 1206)
##   Draws: 4 chains, each with iter = 6000; warmup = 1500; thin = 1;
##          total post-warmup draws = 18000
##
## Multilevel Hyperparameters:
## ~subID (Number of levels: 67)
##                                                 Estimate Est.Error l-95% CI
## sd(Intercept)                                       0.23      0.02     0.19
## sd(expected1)                                       0.03      0.01     0.02
## sd(phase1)                                          0.08      0.01     0.06
## sd(phase2)                                          0.03      0.01     0.01
## sd(difficulty1)                                     0.02      0.01     0.00
## sd(difficulty2)                                     0.02      0.01     0.01
## sd(expected1:phase1)                                0.04      0.01     0.02
## sd(expected1:phase2)                                0.02      0.01     0.00
## sd(phase1:difficulty1)                              0.04      0.01     0.02
## sd(phase2:difficulty1)                              0.02      0.01     0.00
## sd(phase1:difficulty2)                              0.03      0.01     0.01
## sd(phase2:difficulty2)                              0.03      0.01     0.01
## sd(expected1:difficulty1)                           0.01      0.01     0.00
## sd(expected1:difficulty2)                           0.03      0.01     0.01
## cor(Intercept,expected1)                           -0.10      0.15    -0.39
## cor(Intercept,phase1)                               0.01      0.12    -0.22
## cor(expected1,phase1)                               0.24      0.15    -0.07
## cor(Intercept,phase2)                              -0.10      0.17    -0.42
## cor(expected1,phase2)                               0.16      0.20    -0.25
## cor(phase1,phase2)                                 -0.08      0.18    -0.42
## cor(Intercept,difficulty1)                         -0.01      0.19    -0.39
## cor(expected1,difficulty1)                         -0.05      0.22    -0.47
## cor(phase1,difficulty1)                            -0.07      0.20    -0.45
## cor(phase2,difficulty1)                            -0.10      0.23    -0.52
## cor(Intercept,difficulty2)                         -0.07      0.18    -0.42
## cor(expected1,difficulty2)                         -0.17      0.20    -0.54
## cor(phase1,difficulty2)                            -0.33      0.18    -0.65
## cor(phase2,difficulty2)                            -0.12      0.21    -0.52
## cor(difficulty1,difficulty2)                       -0.14      0.24    -0.57
## cor(Intercept,expected1:phase1)                    -0.02      0.15    -0.30
## cor(expected1,expected1:phase1)                     0.11      0.18    -0.25
## cor(phase1,expected1:phase1)                       -0.33      0.15    -0.61
## cor(phase2,expected1:phase1)                       -0.02      0.20    -0.41
```

```
## cor(difficulty1,expected1:phase1)                  -0.12      0.22    -0.53
## cor(difficulty2,expected1:phase1)                   0.12      0.20    -0.29
## cor(Intercept,expected1:phase2)                     0.19      0.19    -0.19
## cor(expected1,expected1:phase2)                    -0.16      0.21    -0.55
## cor(phase1,expected1:phase2)                       -0.12      0.19    -0.49
## cor(phase2,expected1:phase2)                       -0.37      0.22    -0.72
## cor(difficulty1,expected1:phase2)                   0.02      0.23    -0.42
## cor(difficulty2,expected1:phase2)                   0.19      0.22    -0.26
## cor(expected1:phase1,expected1:phase2)             -0.00      0.21    -0.40
## cor(Intercept,phase1:difficulty1)                  -0.32      0.15    -0.60
## cor(expected1,phase1:difficulty1)                  -0.19      0.19    -0.54
## cor(phase1,phase1:difficulty1)                      0.06      0.17    -0.27
## cor(phase2,phase1:difficulty1)                     -0.08      0.21    -0.47
## cor(difficulty1,phase1:difficulty1)                 0.11      0.22    -0.33
## cor(difficulty2,phase1:difficulty1)                -0.02      0.21    -0.42
## cor(expected1:phase1,phase1:difficulty1)           -0.24      0.19    -0.58
## cor(expected1:phase2,phase1:difficulty1)           -0.08      0.21    -0.48
## cor(Intercept,phase2:difficulty1)                  -0.20      0.20    -0.55
## cor(expected1,phase2:difficulty1)                  -0.16      0.22    -0.56
## cor(phase1,phase2:difficulty1)                     -0.16      0.20    -0.53
## cor(phase2,phase2:difficulty1)                     -0.14      0.22    -0.55
## cor(difficulty1,phase2:difficulty1)                -0.02      0.23    -0.47
## cor(difficulty2,phase2:difficulty1)                 0.20      0.22    -0.27
## cor(expected1:phase1,phase2:difficulty1)            0.09      0.21    -0.34
## cor(expected1:phase2,phase2:difficulty1)            0.14      0.23    -0.33
## cor(phase1:difficulty1,phase2:difficulty1)          0.05      0.22    -0.37
## cor(Intercept,phase1:difficulty2)                   0.17      0.18    -0.21
## cor(expected1,phase1:difficulty2)                   0.17      0.21    -0.26
## cor(phase1,phase1:difficulty2)                      0.04      0.19    -0.33
## cor(phase2,phase1:difficulty2)                      0.17      0.22    -0.29
## cor(difficulty1,phase1:difficulty2)                -0.11      0.23    -0.53
## cor(difficulty2,phase1:difficulty2)                -0.15      0.21    -0.55
## cor(expected1:phase1,phase1:difficulty2)           -0.03      0.21    -0.43
## cor(expected1:phase2,phase1:difficulty2)           -0.14      0.22    -0.54
## cor(phase1:difficulty1,phase1:difficulty2)         -0.20      0.22    -0.58
## cor(phase2:difficulty1,phase1:difficulty2)         -0.16      0.23    -0.58
## cor(Intercept,phase2:difficulty2)                   0.11      0.17    -0.24
## cor(expected1,phase2:difficulty2)                   0.22      0.20    -0.18
## cor(phase1,phase2:difficulty2)                     -0.05      0.18    -0.40
## cor(phase2,phase2:difficulty2)                      0.09      0.21    -0.32
## cor(difficulty1,phase2:difficulty2)                 0.03      0.23    -0.42
## cor(difficulty2,phase2:difficulty2)                -0.09      0.21    -0.50
## cor(expected1:phase1,phase2:difficulty2)            0.20      0.20    -0.20
## cor(expected1:phase2,phase2:difficulty2)           -0.05      0.22    -0.46
## cor(phase1:difficulty1,phase2:difficulty2)         -0.29      0.20    -0.64
## cor(phase2:difficulty1,phase2:difficulty2)         -0.25      0.24    -0.66
## cor(phase1:difficulty2,phase2:difficulty2)          0.08      0.22    -0.34
## cor(Intercept,expected1:difficulty1)                0.03      0.22    -0.40
## cor(expected1,expected1:difficulty1)                0.08      0.23    -0.37
## cor(phase1,expected1:difficulty1)                   0.04      0.22    -0.39
## cor(phase2,expected1:difficulty1)                   0.05      0.23    -0.40
## cor(difficulty1,expected1:difficulty1)             -0.18      0.26    -0.63
## cor(difficulty2,expected1:difficulty1)             -0.01      0.23    -0.46
## cor(expected1:phase1,expected1:difficulty1)         0.14      0.24    -0.34
```

```
## cor(expected1:phase2,expected1:difficulty1)         -0.00      0.23    -0.46
## cor(phase1:difficulty1,expected1:difficulty1)       -0.04      0.23    -0.48
## cor(phase2:difficulty1,expected1:difficulty1)        0.07      0.24    -0.41
## cor(phase1:difficulty2,expected1:difficulty1)        0.01      0.23    -0.43
## cor(phase2:difficulty2,expected1:difficulty1)        0.01      0.23    -0.44
## cor(Intercept,expected1:difficulty2)                 0.05      0.16    -0.26
## cor(expected1,expected1:difficulty2)                 0.22      0.19    -0.16
## cor(phase1,expected1:difficulty2)                    0.41      0.16     0.07
## cor(phase2,expected1:difficulty2)                    0.10      0.20    -0.30
## cor(difficulty1,expected1:difficulty2)               0.01      0.23    -0.43
## cor(difficulty2,expected1:difficulty2)              -0.45      0.19    -0.76
## cor(expected1:phase1,expected1:difficulty2)         -0.10      0.19    -0.46
## cor(expected1:phase2,expected1:difficulty2)         -0.14      0.21    -0.54
## cor(phase1:difficulty1,expected1:difficulty2)       -0.01      0.20    -0.41
## cor(phase2:difficulty1,expected1:difficulty2)       -0.15      0.22    -0.55
## cor(phase1:difficulty2,expected1:difficulty2)        0.11      0.21    -0.31
## cor(phase2:difficulty2,expected1:difficulty2)        0.15      0.20    -0.26
## cor(expected1:difficulty1,expected1:difficulty2)    -0.03      0.23    -0.46
##                                                  u-95% CI Rhat Bulk_ESS
## sd(Intercept)                                        0.27 1.00     2450
## sd(expected1)                                        0.04 1.00     7319
## sd(phase1)                                           0.10 1.00     7128
## sd(phase2)                                           0.04 1.00     2757
## sd(difficulty1)                                      0.04 1.00     2622
## sd(difficulty2)                                      0.03 1.00     2914
## sd(expected1:phase1)                                 0.05 1.00     7406
## sd(expected1:phase2)                                 0.04 1.00     3607
## sd(phase1:difficulty1)                               0.06 1.00     6572
## sd(phase2:difficulty1)                               0.04 1.00     3574
## sd(phase1:difficulty2)                               0.05 1.00     3336
## sd(phase2:difficulty2)                               0.05 1.00     3663
## sd(expected1:difficulty1)                            0.03 1.00     3881
## sd(expected1:difficulty2)                            0.04 1.00     5933
## cor(Intercept,expected1)                             0.19 1.00    14999
## cor(Intercept,phase1)                                0.24 1.00     7084
## cor(expected1,phase1)                                0.53 1.00     2921
## cor(Intercept,phase2)                                0.24 1.00    17645
## cor(expected1,phase2)                                0.53 1.00     9920
## cor(phase1,phase2)                                   0.29 1.00    12322
## cor(Intercept,difficulty1)                           0.38 1.00    19542
## cor(expected1,difficulty1)                           0.37 1.00    14194
## cor(phase1,difficulty1)                              0.34 1.00    17806
## cor(phase2,difficulty1)                              0.36 1.00    11516
## cor(Intercept,difficulty2)                           0.28 1.00    20071
## cor(expected1,difficulty2)                           0.24 1.00     9899
## cor(phase1,difficulty2)                              0.06 1.00    12917
## cor(phase2,difficulty2)                              0.32 1.00    12327
## cor(difficulty1,difficulty2)                         0.35 1.00     6429
## cor(Intercept,expected1:phase1)                      0.27 1.00    15080
## cor(expected1,expected1:phase1)                      0.46 1.00     7191
## cor(phase1,expected1:phase1)                        -0.02 1.00    13157
## cor(phase2,expected1:phase1)                         0.38 1.00     7457
## cor(difficulty1,expected1:phase1)                    0.33 1.00     5568
## cor(difficulty2,expected1:phase1)                    0.49 1.00     7811
```

```
## cor(Intercept,expected1:phase2)                       0.53 1.00    18788
## cor(expected1,expected1:phase2)                       0.26 1.00    12115
## cor(phase1,expected1:phase2)                          0.26 1.00    15016
## cor(phase2,expected1:phase2)                          0.13 1.00     5573
## cor(difficulty1,expected1:phase2)                     0.46 1.00    15084
## cor(difficulty2,expected1:phase2)                     0.59 1.00    10441
## cor(expected1:phase1,expected1:phase2)                0.42 1.00    16150
## cor(Intercept,phase1:difficulty1)                     0.01 1.00    16667
## cor(expected1,phase1:difficulty1)                     0.21 1.00    11003
## cor(phase1,phase1:difficulty1)                        0.39 1.00    15879
## cor(phase2,phase1:difficulty1)                        0.34 1.00    10200
## cor(difficulty1,phase1:difficulty1)                   0.52 1.00     8626
## cor(difficulty2,phase1:difficulty1)                   0.40 1.00     9251
## cor(expected1:phase1,phase1:difficulty1)              0.15 1.00    12967
## cor(expected1:phase2,phase1:difficulty1)              0.35 1.00    10985
## cor(Intercept,phase2:difficulty1)                     0.21 1.00    14999
## cor(expected1,phase2:difficulty1)                     0.28 1.00    13081
## cor(phase1,phase2:difficulty1)                        0.26 1.00    18103
## cor(phase2,phase2:difficulty1)                        0.31 1.00    11693
## cor(difficulty1,phase2:difficulty1)                   0.43 1.00    12559
## cor(difficulty2,phase2:difficulty1)                   0.60 1.00     9945
## cor(expected1:phase1,phase2:difficulty1)              0.49 1.00    17305
## cor(expected1:phase2,phase2:difficulty1)              0.55 1.00    11666
## cor(phase1:difficulty1,phase2:difficulty1)            0.48 1.00    15662
## cor(Intercept,phase1:difficulty2)                     0.51 1.00    17609
## cor(expected1,phase1:difficulty2)                     0.55 1.00    10914
## cor(phase1,phase1:difficulty2)                        0.41 1.00    18467
## cor(phase2,phase1:difficulty2)                        0.56 1.00     9752
## cor(difficulty1,phase1:difficulty2)                   0.35 1.00     8462
## cor(difficulty2,phase1:difficulty2)                   0.28 1.00    10370
## cor(expected1:phase1,phase1:difficulty2)              0.38 1.00    15493
## cor(expected1:phase2,phase1:difficulty2)              0.31 1.00    10789
## cor(phase1:difficulty1,phase1:difficulty2)            0.26 1.00     9872
## cor(phase2:difficulty1,phase1:difficulty2)            0.31 1.00    10018
## cor(Intercept,phase2:difficulty2)                     0.43 1.00    16756
## cor(expected1,phase2:difficulty2)                     0.58 1.00    10244
## cor(phase1,phase2:difficulty2)                        0.31 1.00    18573
## cor(phase2,phase2:difficulty2)                        0.49 1.00    10111
## cor(difficulty1,phase2:difficulty2)                   0.46 1.00     8265
## cor(difficulty2,phase2:difficulty2)                   0.33 1.00     9317
## cor(expected1:phase1,phase2:difficulty2)              0.57 1.00    13392
## cor(expected1:phase2,phase2:difficulty2)              0.38 1.00    10727
## cor(phase1:difficulty1,phase2:difficulty2)            0.13 1.00    11223
## cor(phase2:difficulty1,phase2:difficulty2)            0.26 1.00     5626
## cor(phase1:difficulty2,phase2:difficulty2)            0.50 1.00    11869
## cor(Intercept,expected1:difficulty1)                  0.45 1.00    24414
## cor(expected1,expected1:difficulty1)                  0.51 1.00    17565
## cor(phase1,expected1:difficulty1)                     0.46 1.00    22856
## cor(phase2,expected1:difficulty1)                     0.49 1.00    16404
## cor(difficulty1,expected1:difficulty1)                0.35 1.00     6074
## cor(difficulty2,expected1:difficulty1)                0.45 1.00    14129
## cor(expected1:phase1,expected1:difficulty1)           0.56 1.00    13137
## cor(expected1:phase2,expected1:difficulty1)           0.45 1.00    18155
## cor(phase1:difficulty1,expected1:difficulty1)         0.40 1.00    17735
```

```
## cor(phase2:difficulty1,expected1:difficulty1)        0.52 1.00    12251
## cor(phase1:difficulty2,expected1:difficulty1)        0.46 1.00    15268
## cor(phase2:difficulty2,expected1:difficulty1)        0.46 1.00    16391
## cor(Intercept,expected1:difficulty2)                 0.36 1.00    19680
## cor(expected1,expected1:difficulty2)                 0.58 1.00     8800
## cor(phase1,expected1:difficulty2)                    0.70 1.00    13856
## cor(phase2,expected1:difficulty2)                    0.49 1.00    11524
## cor(difficulty1,expected1:difficulty2)               0.45 1.00     7792
## cor(difficulty2,expected1:difficulty2)               0.00 1.00     4649
## cor(expected1:phase1,expected1:difficulty2)          0.28 1.00    15185
## cor(expected1:phase2,expected1:difficulty2)          0.28 1.00    13428
## cor(phase1:difficulty1,expected1:difficulty2)        0.37 1.00    14012
## cor(phase2:difficulty1,expected1:difficulty2)        0.29 1.00    11321
## cor(phase1:difficulty2,expected1:difficulty2)        0.51 1.00    13720
## cor(phase2:difficulty2,expected1:difficulty2)        0.52 1.00    11864
## cor(expected1:difficulty1,expected1:difficulty2)     0.44 1.00    13520
##                                                  Tail_ESS
## sd(Intercept)                                        5016
## sd(expected1)                                        9081
## sd(phase1)                                          11503
## sd(phase2)                                           2498
## sd(difficulty1)                                      4947
## sd(difficulty2)                                      1914
## sd(expected1:phase1)                                 8361
## sd(expected1:phase2)                                 2958
## sd(phase1:difficulty1)                               5613
## sd(phase2:difficulty1)                               4931
## sd(phase1:difficulty2)                               2719
## sd(phase2:difficulty2)                               3283
## sd(expected1:difficulty1)                            6961
## sd(expected1:difficulty2)                            4747
## cor(Intercept,expected1)                            14393
## cor(Intercept,phase1)                               10477
## cor(expected1,phase1)                                6426
## cor(Intercept,phase2)                               12525
## cor(expected1,phase2)                               11415
## cor(phase1,phase2)                                  11896
## cor(Intercept,difficulty1)                          12851
## cor(expected1,difficulty1)                          13695
## cor(phase1,difficulty1)                             12891
## cor(phase2,difficulty1)                             13377
## cor(Intercept,difficulty2)                          11806
## cor(expected1,difficulty2)                          10894
## cor(phase1,difficulty2)                              8506
## cor(phase2,difficulty2)                             12326
## cor(difficulty1,difficulty2)                        10874
## cor(Intercept,expected1:phase1)                     14481
## cor(expected1,expected1:phase1)                     11078
## cor(phase1,expected1:phase1)                        13962
## cor(phase2,expected1:phase1)                        10636
## cor(difficulty1,expected1:phase1)                    7918
## cor(difficulty2,expected1:phase1)                   12628
## cor(Intercept,expected1:phase2)                     12433
## cor(expected1,expected1:phase2)                     12979
```

```
## cor(phase1,expected1:phase2)                          12091
## cor(phase2,expected1:phase2)                           7264
## cor(difficulty1,expected1:phase2)                     13885
## cor(difficulty2,expected1:phase2)                     10977
## cor(expected1:phase1,expected1:phase2)                14439
## cor(Intercept,phase1:difficulty1)                     14537
## cor(expected1,phase1:difficulty1)                     12951
## cor(phase1,phase1:difficulty1)                        14896
## cor(phase2,phase1:difficulty1)                        12718
## cor(difficulty1,phase1:difficulty1)                   10819
## cor(difficulty2,phase1:difficulty1)                   12692
## cor(expected1:phase1,phase1:difficulty1)              13450
## cor(expected1:phase2,phase1:difficulty1)              13657
## cor(Intercept,phase2:difficulty1)                     10857
## cor(expected1,phase2:difficulty1)                     11534
## cor(phase1,phase2:difficulty1)                        12650
## cor(phase2,phase2:difficulty1)                        12803
## cor(difficulty1,phase2:difficulty1)                   12554
## cor(difficulty2,phase2:difficulty1)                   12538
## cor(expected1:phase1,phase2:difficulty1)              14537
## cor(expected1:phase2,phase2:difficulty1)              13880
## cor(phase1:difficulty1,phase2:difficulty1)            14239
## cor(Intercept,phase1:difficulty2)                     12244
## cor(expected1,phase1:difficulty2)                     12986
## cor(phase1,phase1:difficulty2)                        13248
## cor(phase2,phase1:difficulty2)                        10715
## cor(difficulty1,phase1:difficulty2)                   11870
## cor(difficulty2,phase1:difficulty2)                   13191
## cor(expected1:phase1,phase1:difficulty2)              15347
## cor(expected1:phase2,phase1:difficulty2)              12065
## cor(phase1:difficulty1,phase1:difficulty2)            10030
## cor(phase2:difficulty1,phase1:difficulty2)            12999
## cor(Intercept,phase2:difficulty2)                     12159
## cor(expected1,phase2:difficulty2)                     12320
## cor(phase1,phase2:difficulty2)                        13873
## cor(phase2,phase2:difficulty2)                        12461
## cor(difficulty1,phase2:difficulty2)                   10967
## cor(difficulty2,phase2:difficulty2)                   13014
## cor(expected1:phase1,phase2:difficulty2)              14512
## cor(expected1:phase2,phase2:difficulty2)              13454
## cor(phase1:difficulty1,phase2:difficulty2)            13134
## cor(phase2:difficulty1,phase2:difficulty2)            10452
## cor(phase1:difficulty2,phase2:difficulty2)            13426
## cor(Intercept,expected1:difficulty1)                  12088
## cor(expected1,expected1:difficulty1)                  13922
## cor(phase1,expected1:difficulty1)                     13968
## cor(phase2,expected1:difficulty1)                     13155
## cor(difficulty1,expected1:difficulty1)                12675
## cor(difficulty2,expected1:difficulty1)                13798
## cor(expected1:phase1,expected1:difficulty1)           12985
## cor(expected1:phase2,expected1:difficulty1)           15806
## cor(phase1:difficulty1,expected1:difficulty1)         15233
## cor(phase2:difficulty1,expected1:difficulty1)         14202
## cor(phase1:difficulty2,expected1:difficulty1)         15404
```

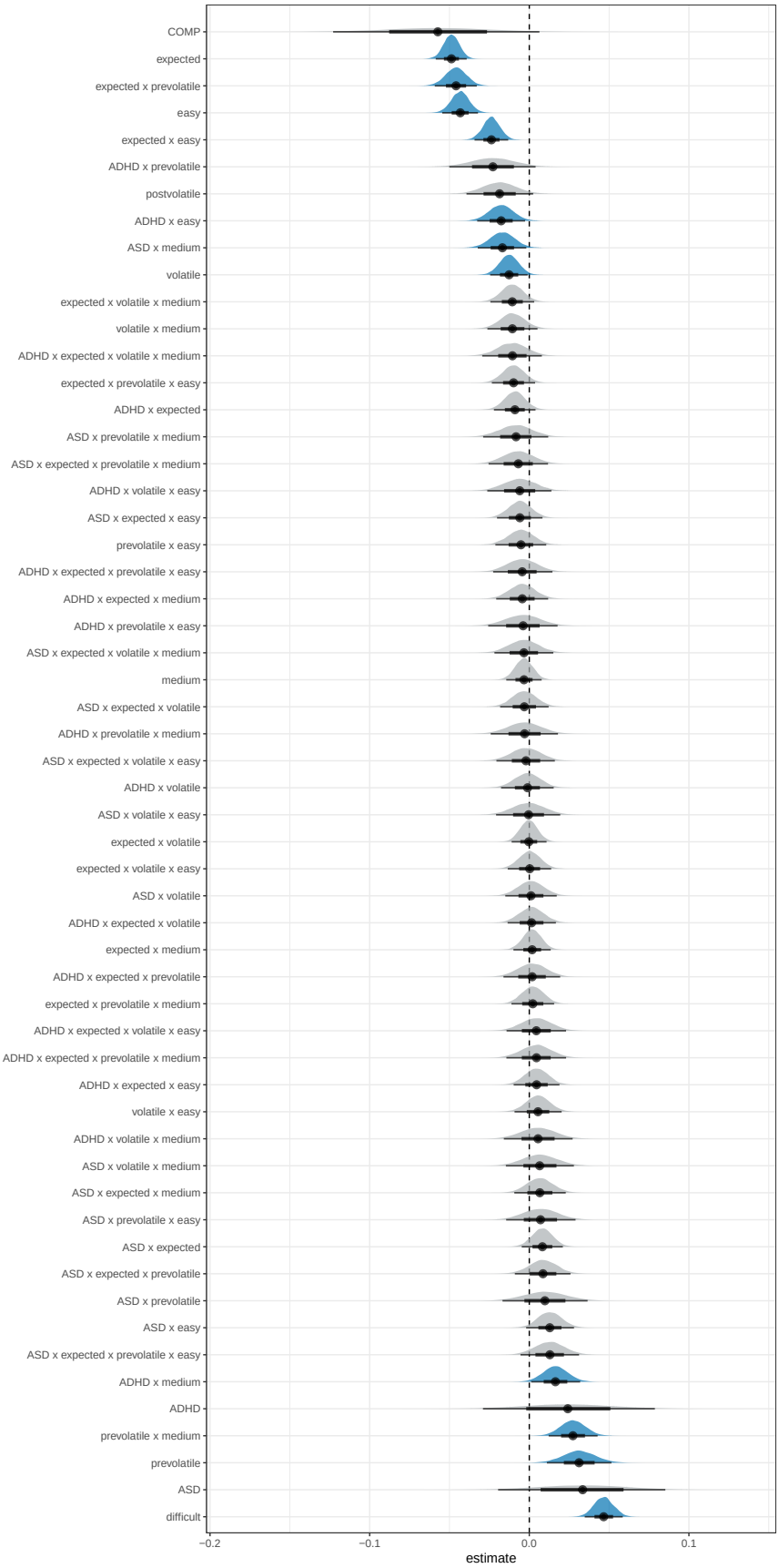
```
## cor(phase2:difficulty2,expected1:difficulty1)       15431
## cor(Intercept,expected1:difficulty2)                12544
## cor(expected1,expected1:difficulty2)                11890
## cor(phase1,expected1:difficulty2)                   12165
## cor(phase2,expected1:difficulty2)                   12681
## cor(difficulty1,expected1:difficulty2)              12398
## cor(difficulty2,expected1:difficulty2)               4719
## cor(expected1:phase1,expected1:difficulty2)         15375
## cor(expected1:phase2,expected1:difficulty2)         13440
## cor(phase1:difficulty1,expected1:difficulty2)       14895
## cor(phase2:difficulty1,expected1:difficulty2)       13460
## cor(phase1:difficulty2,expected1:difficulty2)       14890
## cor(phase2:difficulty2,expected1:difficulty2)       14356
## cor(expected1:difficulty1,expected1:difficulty2)    15348
##
## Regression Coefficients:
##                                          Estimate Est.Error l-95% CI u-95% CI
## Intercept                                    5.92      0.04     5.84     6.01
## diagnosis1                                   0.02      0.03    -0.03     0.08
## diagnosis2                                   0.03      0.03    -0.02     0.09
## expected1                                   -0.05      0.00    -0.06    -0.04
## phase1                                       0.03      0.01     0.01     0.05
## phase2                                      -0.01      0.01    -0.02    -0.00
## difficulty1                                 -0.04      0.01    -0.05    -0.03
## difficulty2                                 -0.00      0.01    -0.01     0.01
## diagnosis1:expected1                        -0.01      0.01    -0.02     0.00
## diagnosis2:expected1                         0.01      0.01    -0.00     0.02
## diagnosis1:phase1                           -0.02      0.01    -0.05     0.00
## diagnosis2:phase1                            0.01      0.01    -0.02     0.04
## diagnosis1:phase2                           -0.00      0.01    -0.02     0.02
## diagnosis2:phase2                            0.00      0.01    -0.02     0.02
## expected1:phase1                            -0.05      0.01    -0.06    -0.03
## expected1:phase2                            -0.00      0.01    -0.01     0.01
## diagnosis1:difficulty1                      -0.02      0.01    -0.03    -0.00
## diagnosis2:difficulty1                       0.01      0.01    -0.00     0.03
## diagnosis1:difficulty2                       0.02      0.01     0.00     0.03
## diagnosis2:difficulty2                      -0.02      0.01    -0.03    -0.00
## expected1:difficulty1                       -0.02      0.01    -0.03    -0.01
## expected1:difficulty2                        0.00      0.01    -0.01     0.01
## phase1:difficulty1                          -0.01      0.01    -0.02     0.01
## phase2:difficulty1                           0.01      0.01    -0.01     0.02
## phase1:difficulty2                           0.03      0.01     0.01     0.04
## phase2:difficulty2                          -0.01      0.01    -0.03     0.01
## diagnosis1:expected1:phase1                  0.00      0.01    -0.02     0.02
## diagnosis2:expected1:phase1                  0.01      0.01    -0.01     0.03
## diagnosis1:expected1:phase2                  0.00      0.01    -0.01     0.02
## diagnosis2:expected1:phase2                 -0.00      0.01    -0.02     0.01
## diagnosis1:expected1:difficulty1             0.00      0.01    -0.01     0.02
## diagnosis2:expected1:difficulty1            -0.01      0.01    -0.02     0.01
## diagnosis1:expected1:difficulty2            -0.00      0.01    -0.02     0.01
## diagnosis2:expected1:difficulty2             0.01      0.01    -0.01     0.02
## diagnosis1:phase1:difficulty1               -0.00      0.01    -0.03     0.02
## diagnosis2:phase1:difficulty1                0.01      0.01    -0.01     0.03
## diagnosis1:phase2:difficulty1               -0.01      0.01    -0.03     0.01
```

```
## diagnosis2:phase2:difficulty1              -0.00      0.01    -0.02     0.02
## diagnosis1:phase1:difficulty2              -0.00      0.01    -0.02     0.02
## diagnosis2:phase1:difficulty2              -0.01      0.01    -0.03     0.01
## diagnosis1:phase2:difficulty2               0.01      0.01    -0.02     0.03
## diagnosis2:phase2:difficulty2               0.01      0.01    -0.01     0.03
## expected1:phase1:difficulty1               -0.01      0.01    -0.02     0.00
## expected1:phase2:difficulty1                0.00      0.01    -0.01     0.01
## expected1:phase1:difficulty2                0.00      0.01    -0.01     0.02
## expected1:phase2:difficulty2               -0.01      0.01    -0.02     0.00
## diagnosis1:expected1:phase1:difficulty1    -0.00      0.01    -0.02     0.01
## diagnosis2:expected1:phase1:difficulty1     0.01      0.01    -0.01     0.03
## diagnosis1:expected1:phase2:difficulty1     0.00      0.01    -0.01     0.02
## diagnosis2:expected1:phase2:difficulty1    -0.00      0.01    -0.02     0.02
## diagnosis1:expected1:phase1:difficulty2     0.00      0.01    -0.01     0.02
## diagnosis2:expected1:phase1:difficulty2    -0.01      0.01    -0.03     0.01
## diagnosis1:expected1:phase2:difficulty2    -0.01      0.01    -0.03     0.01
## diagnosis2:expected1:phase2:difficulty2    -0.00      0.01    -0.02     0.02
##                                          Rhat Bulk_ESS Tail_ESS
## Intercept                                1.00     1944     4366
## diagnosis1                               1.00     1694     4167
## diagnosis2                               1.00     1985     4257
## expected1                                1.00    11223    13680
## phase1                                   1.00     6402    10008
## phase2                                   1.00    17082    13815
## difficulty1                              1.00    15932    13737
## difficulty2                              1.00    17550    13763
## diagnosis1:expected1                     1.00    10432    12677
## diagnosis2:expected1                     1.00    10822    12533
## diagnosis1:phase1                        1.00     6304     9796
## diagnosis2:phase1                        1.00     6227     9136
## diagnosis1:phase2                        1.00    13213    12685
## diagnosis2:phase2                        1.00    13020    13107
## expected1:phase1                         1.00    12234    13780
## expected1:phase2                         1.00    18610    14099
## diagnosis1:difficulty1                   1.00    15092    14575
## diagnosis2:difficulty1                   1.00    15778    13939
## diagnosis1:difficulty2                   1.00    14042    14200
## diagnosis2:difficulty2                   1.00    13940    14439
## expected1:difficulty1                    1.00    20360    14030
## expected1:difficulty2                    1.00    15491    14059
## phase1:difficulty1                       1.00    11551    13725
## phase2:difficulty1                       1.00    15511    12798
## phase1:difficulty2                       1.00    14999    14129
## phase2:difficulty2                       1.00    14665    13695
## diagnosis1:expected1:phase1              1.00    11604    12934
## diagnosis2:expected1:phase1              1.00    11911    12959
## diagnosis1:expected1:phase2              1.00    16077    14584
## diagnosis2:expected1:phase2              1.00    15615    14003
## diagnosis1:expected1:difficulty1         1.00    17547    13394
## diagnosis2:expected1:difficulty1         1.00    17111    14290
## diagnosis1:expected1:difficulty2         1.00    12092    12750
## diagnosis2:expected1:difficulty2         1.00    12430    13567
## diagnosis1:phase1:difficulty1            1.00    11244    14140
## diagnosis2:phase1:difficulty1            1.00    12150    13056
```

```
## diagnosis1:phase2:difficulty1           1.00    13386    13815
## diagnosis2:phase2:difficulty1           1.00    13876    14000
## diagnosis1:phase1:difficulty2           1.00    14483    14362
## diagnosis2:phase1:difficulty2           1.00    15152    14519
## diagnosis1:phase2:difficulty2           1.00    12788    13665
## diagnosis2:phase2:difficulty2           1.00    13071    13490
## expected1:phase1:difficulty1            1.00    17610    14991
## expected1:phase2:difficulty1            1.00    17424    14269
## expected1:phase1:difficulty2            1.00    17217    14073
## expected1:phase2:difficulty2            1.00    17715    14782
## diagnosis1:expected1:phase1:difficulty1 1.00    14866    13882
## diagnosis2:expected1:phase1:difficulty1 1.00    15039    14118
## diagnosis1:expected1:phase2:difficulty1 1.00    15210    14445
## diagnosis2:expected1:phase2:difficulty1 1.00    14908    14362
## diagnosis1:expected1:phase1:difficulty2 1.00    15012    13653
## diagnosis2:expected1:phase1:difficulty2 1.00    14887    14069
## diagnosis1:expected1:phase2:difficulty2 1.00    14575    14574
## diagnosis2:expected1:phase2:difficulty2 1.00    14340    14542
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.12      0.01     0.11     0.13 1.00     2893     6974
## ndt     207.00     12.44   180.64   229.38 1.00     6441     9869
##
## 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 group comparisons
df.m.pal = post.draws %>%
  select(starts_with("b_")) %>%
  mutate(
    b_COMP         = - b_diagnosis1 - b_diagnosis2,
    b_postvolatile = - b_phase1 - b_phase2,
    b_difficult    = - b_difficulty1 - b_difficulty2,
    ASD            = b_Intercept + b_diagnosis2,
    ADHD           = b_Intercept + b_diagnosis1,
    COMP           = b_Intercept + b_COMP,
    ASD_expected   = b_Intercept + b_diagnosis2 + b_expected1 +
      `b_diagnosis2:expected1`,
    ASD_unexpected = b_Intercept + b_diagnosis2 - b_expected1 -
      `b_diagnosis2:expected1`,
    ADHD_expected   = b_Intercept + b_diagnosis1 + b_expected1 +
      `b_diagnosis1:expected1`,
    ADHD_unexpected = b_Intercept + b_diagnosis1 - b_expected1 -
      `b_diagnosis1:expected1`,
    COMP_expected   = b_Intercept - b_diagnosis1 - b_diagnosis2 + b_expected1 -
      `b_diagnosis1:expected1` - `b_diagnosis2:expected1`,
    COMP_unexpected   = b_Intercept - b_diagnosis1 - b_diagnosis2 -
      b_expected1 + `b_diagnosis1:expected1` + `b_diagnosis2:expected1`,
    `ADHD-ASD`     = ADHD - ASD,
    `ADHD-COMP`    = ADHD - COMP,
    `ASD-COMP`     = ASD  - COMP,
    h1b = (COMP_unexpected - COMP_expected) - (ASD_unexpected  - ASD_expected),
```

```
    h1c = (COMP_unexpected - COMP_expected) - (ADHD_unexpected - ADHD_expected),
    h1d = `b_diagnosis1:phase2` + 2*`b_diagnosis2:phase2`
  )

# plot the posterior distributions
df.m.pal %>%
  pivot_longer(cols = starts_with("b_"), names_to = "coef", values_to = "estimate") %>%
  subset(!startsWith(coef, "b_Int")) %>%
  mutate(
    coef = substr(coef, 3, nchar(coef)),
    coef = str_replace_all(coef, ":", " x "),
    coef = str_replace_all(coef, "diagnosis1", "ADHD"),
    coef = str_replace_all(coef, "diagnosis2", "ASD"),
    coef = str_replace_all(coef, "expected1", "expected"),
    coef = str_replace_all(coef, "expected2", "unexpected"),
    coef = str_replace_all(coef, "phase1", "prevolatile"),
    coef = str_replace_all(coef, "phase2", "volatile"),
    coef = str_replace_all(coef, "difficulty1", "easy"),
    coef = str_replace_all(coef, "difficulty2", "medium"),
    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(c_dark, c_light)) + theme(legend.position = "none")
```


COMP
expected
expected x prevolatile
easy
expected x easy
ADHD x prevolatile
postvolatile
ADHD x easy
ASD x medium
volatile
expected x volatile x medium
volatile x medium
ADHD x expected x volatile x medium
expected x prevolatile x easy
ADHD x expected
ASD x prevolatile x medium
ASD x expected x prevolatile x medium
ADHD x volatile x easy
ASD x expected x easy
prevolatile x easy
ADHD x expected x prevolatile x easy
ADHD x expected x medium
ADHD x prevolatile x easy
ASD x expected x volatile x medium
medium
ASD x expected x volatile
ADHD x prevolatile x medium
ASD x expected x volatile x easy
ADHD x volatile
ASD x volatile x easy
expected x volatile
expected x volatile x easy
ASD x volatile
ADHD x expected x volatile
expected x medium
ADHD x expected x prevolatile
expected x prevolatile x medium
ADHD x expected x volatile x easy
ADHD x expected x prevolatile x medium
ADHD x expected x easy
volatile x easy
ADHD x volatile x medium
ASD x volatile x medium
ASD x expected x medium
ASD x prevolatile x easy
ASD x expected
ASD x expected x prevolatile
ASD x prevolatile
ASD x easy
ASD x expected x prevolatile x easy
ADHD x medium
ADHD
prevolatile x medium
prevolatile
ASD
difficult
−0.2
−0.1
0.0
0.1
estimate

```
# H1b: COMP(unexp-exp) > ASD(unexp-exp)
h1b = hypothesis(m.pal, "0 < diagnosis1:expected1 + 2*diagnosis2:expected1")
h1b
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(diagnosis1:e... < 0    -0.01      0.01    -0.03     0.01       2.85
##   Post.Prob Star
## 1      0.74
## ---
## 'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 95%;
## for two-sided hypotheses, the value tested against lies outside the 95%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

```
# H1c: COMP(unexp-exp) != ADHD(unexp-exp)
h1c = hypothesis(m.pal, "0 > 2*diagnosis1:expected1 + diagnosis2:expected1",
                 alpha = 0.025)
h1c
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis1... > 0     0.01      0.01    -0.01     0.03       4.13
##   Post.Prob Star
## 1       0.8
## ---
## '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.
```

```
# H1d:COMP(volatile-prevolatile)) < ASD(volatile-prevolatile)
h1d = hypothesis(m.pal, "0 > -diagnosis1:phase1 - 2*diagnosis2:phase1 +
                 diagnosis1:phase2 + 2*diagnosis2:phase2")
h1d
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-diagnosis1:... > 0        0      0.03    -0.06     0.05        0.8
##   Post.Prob Star
## 1      0.44
## ---
## '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.
```

```
# equivalence of the hypotheses
equ = equivalence_test(df.m.pal %>% select(h1b, h1c, h1d))
equ
```

```
## # Test for Practical Equivalence
##
##   ROPE: [-0.10 0.10]
##
## Parameter |       H0 | inside ROPE |       95% HDI
```

```
## ---------------------------------------------------
## h1b       | Accepted |    100.00 % | [-0.03, 0.06]
## h1c       | Accepted |    100.00 % | [-0.07, 0.03]
## h1d       | Accepted |    100.00 % | [-0.03, 0.03]
```

```
## exploration: prevolatile phase

# prevolatile: COMP(unexpected - expected) != ASD(unexpected - expected)
e1.1 = hypothesis(m.pal, "0 < diagnosis1:expected1 +
                  2*diagnosis2:expected1 +
                  diagnosis1:expected1:phase1 +
                  2*diagnosis2:expected1:phase1", alpha = 0.025)
e1.1
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(diagnosis1:e... < 0    -0.03      0.02    -0.07     0.01       9.78
##   Post.Prob Star
## 1      0.91
## ---
## '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.
```

```
# prevolatile: COMP(unexpected - expected) != ADHD(unexpected - expected)
e1.2 = hypothesis(m.pal, "0 < 2*diagnosis1:expected1 +
                  diagnosis2:expected1 +
                  2*diagnosis1:expected1:phase1 +
                  diagnosis2:expected1:phase1", alpha = 0.025)
e1.2
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis1... < 0        0      0.02    -0.04     0.04        1.2
##   Post.Prob Star
## 1      0.55
## ---
## '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.
```

```
# prevolatile: ADHD(unexpected - expected) != ASD(unexpected - expected)
e1.3 = hypothesis(m.pal, "0 < -diagnosis1:expected1 +
                  diagnosis2:expected1 -
                  diagnosis1:expected1:phase1 +
                  diagnosis2:expected1:phase1", alpha = 0.025)
e1.3
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-diagnosis1:... < 0    -0.02      0.02    -0.06     0.01       8.43
##   Post.Prob Star
## 1      0.89
## ---
## 'CI': 95%-CI for one-sided and 97.5%-CI for two-sided hypotheses.
```

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

```
## exploration: task effects

# expected versus unexpected
e2.1 = hypothesis(m.pal, "0 > expected1", alpha = 0.025)
e2.1
```

```
## Hypothesis Tests for class b:
##            Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob
## 1 (0)-(expected1) > 0     0.05         0     0.04     0.06        Inf         1
##   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.
```

```
# volatile versus prevolatile
e2.2 = hypothesis(m.pal, "0 < phase1 + 2*phase2", alpha = 0.025)
e2.2
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(phase1+2*pha... < 0    -0.01      0.01    -0.03     0.02       2.01
##   Post.Prob Star
## 1      0.67
## ---
## '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.
```

```
# prevolatile versus postvolatile
e2.3 = hypothesis(m.pal, "0 < 2*phase1 + phase2", alpha = 0.025)
e2.3
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*phase1+pha... < 0    -0.05      0.02    -0.09    -0.01     147.76
##   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.
```

```
# volatile versus postvolatile
e2.4 = hypothesis(m.pal, "0 < phase1 - phase2", alpha = 0.025)
e2.4
```

```
## Hypothesis Tests for class b:
##                Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(phase1-phase2) < 0    -0.04      0.01    -0.07    -0.02       4499
```

```
##   Post.Prob Star
## 1         1    *
## ---
## 'CI': 95%-CI for one-sided and 97.5%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 97.5%;
## for two-sided hypotheses, the value tested against lies outside the 97.5%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

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

# calculate our difference columns
df.ms = df.ms %>%
  mutate(
    COMP_expected     = rowMeans(across(matches("COMP_.*_expected_.*"))),
    COMP_unexpected   = rowMeans(across(matches("COMP_.*_unexpected_.*"))),
    ADHD_expected     = rowMeans(across(matches("ADHD_.*_expected_.*"))),
    ADHD_unexpected   = rowMeans(across(matches("ADHD_.*_unexpected_.*"))),
    ASD_expected      = rowMeans(across(matches("ASD_.*_expected_.*"))),
    ASD_unexpected    = rowMeans(across(matches("ASD_.*_unexpected_.*"))),
    COMP_unexp_exp    = COMP_unexpected - COMP_expected,
    ADHD_unexp_exp    = ADHD_unexpected - ADHD_expected,
    ASD_unexp_exp     = ASD_unexpected  - ASD_expected,
    h1b_COMPvASD_exp  = COMP_unexp_exp - ASD_unexp_exp,
    h1c_COMPvADHD_exp = COMP_unexp_exp - ADHD_unexp_exp,
    ASD_prevolatile   = rowMeans(across(matches("ASD_prevolatile_.*"))),
    ASD_volatile      = rowMeans(across(matches("ASD_volatile_.*"))),
    COMP_prevolatile  = rowMeans(across(matches("COMP_prevolatile_.*"))),
    COMP_volatile     = rowMeans(across(matches("COMP_volatile_.*"))),
    h1d_ASDvCOMP_volvpre =
      (ASD_volatile - ASD_prevolatile) - (COMP_volatile - COMP_prevolatile),
    COMP_pre_unexp    = rowMeans(across(matches("COMP_prevolatile_unexpected_.*"))),
    COMP_pre_exp      = rowMeans(across(matches("COMP_prevolatile_expected_.*"))),
    ADHD_pre_unexp    = rowMeans(across(matches("ADHD_prevolatile_unexpected_.*"))),
    ADHD_pre_exp      = rowMeans(across(matches("ADHD_prevolatile_expected_.*"))),
    ASD_pre_unexp     = rowMeans(across(matches("ASD_prevolatile_unexpected_.*"))),
    ASD_pre_exp       = rowMeans(across(matches("ASD_prevolatile_expected_.*"))),
    e11_pre_COMPvASD_exp =
      (COMP_pre_unexp - COMP_pre_exp) - (ASD_pre_unexp - ASD_pre_exp),
    e12_pre_COMPvADHD_exp =
      (COMP_pre_unexp - COMP_pre_exp) - (ADHD_pre_unexp - ADHD_pre_exp),
    e13_pre_ADHDvASD_exp =
      (ADHD_pre_unexp - ADHD_pre_exp) - (ASD_pre_unexp - ASD_pre_exp),
    e21_exp           = rowMeans(across(matches(".*_unexpected_.*"))) -
```

```
      rowMeans(across(matches(".*_expected_.*"))),
    e22_volvpre       = rowMeans(across(matches(".*_volatile_.*"))) -
      rowMeans(across(matches(".*_prevolatile_.*"))),
    e23_prevpost      = rowMeans(across(matches(".*_prevolatile_.*"))) -
      rowMeans(across(matches(".*_postvolatile_.*"))),
    e24_volvpost      = rowMeans(across(matches(".*_volatile_.*"))) -
      rowMeans(across(matches(".*_postvolatile_.*"))),
    e25_diffvmed      = rowMeans(across(matches(".*_difficult"))) -
      rowMeans(across(matches(".*_medium"))),
    e25_medveasy      = rowMeans(across(matches(".*_medium"))) -
      rowMeans(across(matches(".*_easy"))),
    e25_diffveasy     = rowMeans(across(matches(".*_difficult"))) -
      rowMeans(across(matches(".*_easy")))
  )

lapply(df.ms %>% select(starts_with("e") | starts_with("h")), ci)
```

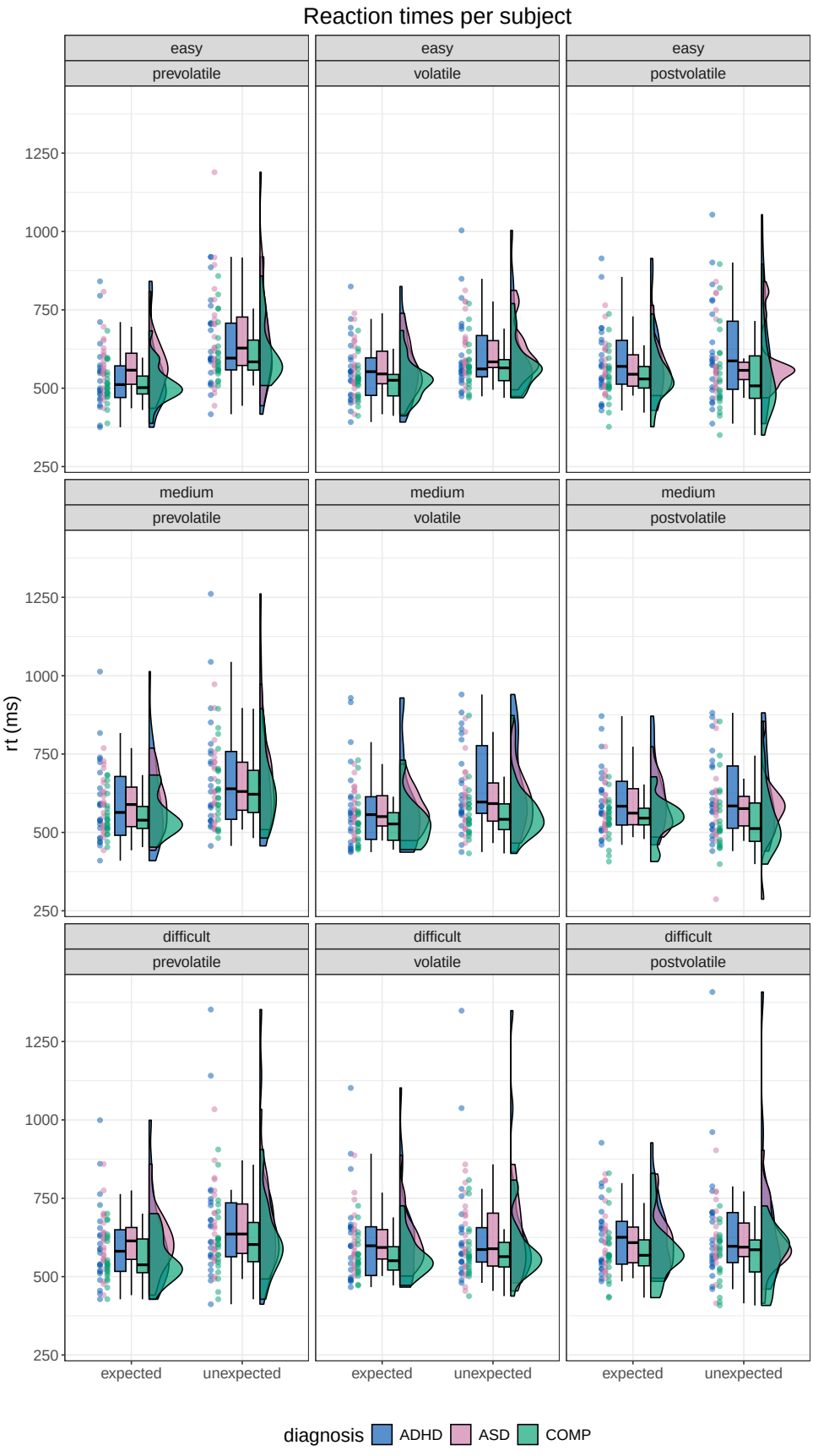
```
## $e11_pre_COMPvASD_exp
## 95% ETI: [-17.67, 46.31]
##
## $e12_pre_COMPvADHD_exp
## 95% ETI: [-33.85, 30.82]
##
## $e13_pre_ADHDvASD_exp
## 95% ETI: [-15.27, 47.68]
##
## $e21_exp
## 95% ETI: [29.75, 44.48]
##
## $e22_volvpre
## 95% ETI: [-28.10, -8.21]
##
## $e23_prevpost
## 95% ETI: [5.18, 35.51]
##
## $e24_volvpost
## 95% ETI: [-7.89, 12.41]
##
## $e25_diffvmed
## 95% ETI: [10.73, 26.31]
##
## $e25_medveasy
## 95% ETI: [7.44, 21.32]
##
## $e25_diffveasy
## 95% ETI: [25.25, 40.64]
##
## $h1b_COMPvASD_exp
## 95% ETI: [-14.27, 20.28]
##
## $h1c_COMPvADHD_exp
## 95% ETI: [-27.02, 8.14]
##
## $h1d_ASDvCOMP_volvpre
```

```
## 95% ETI: [-23.19, 24.86]
```

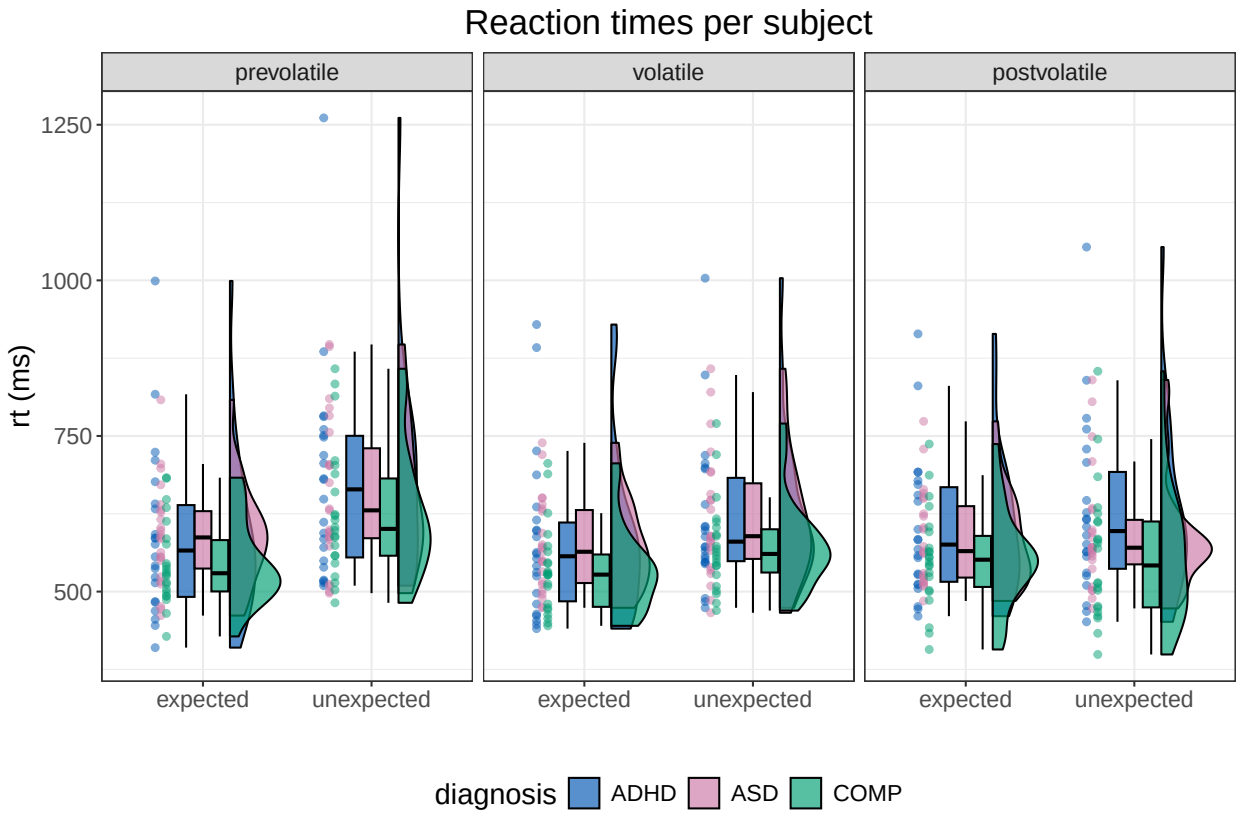
[!MISSING]

Plots

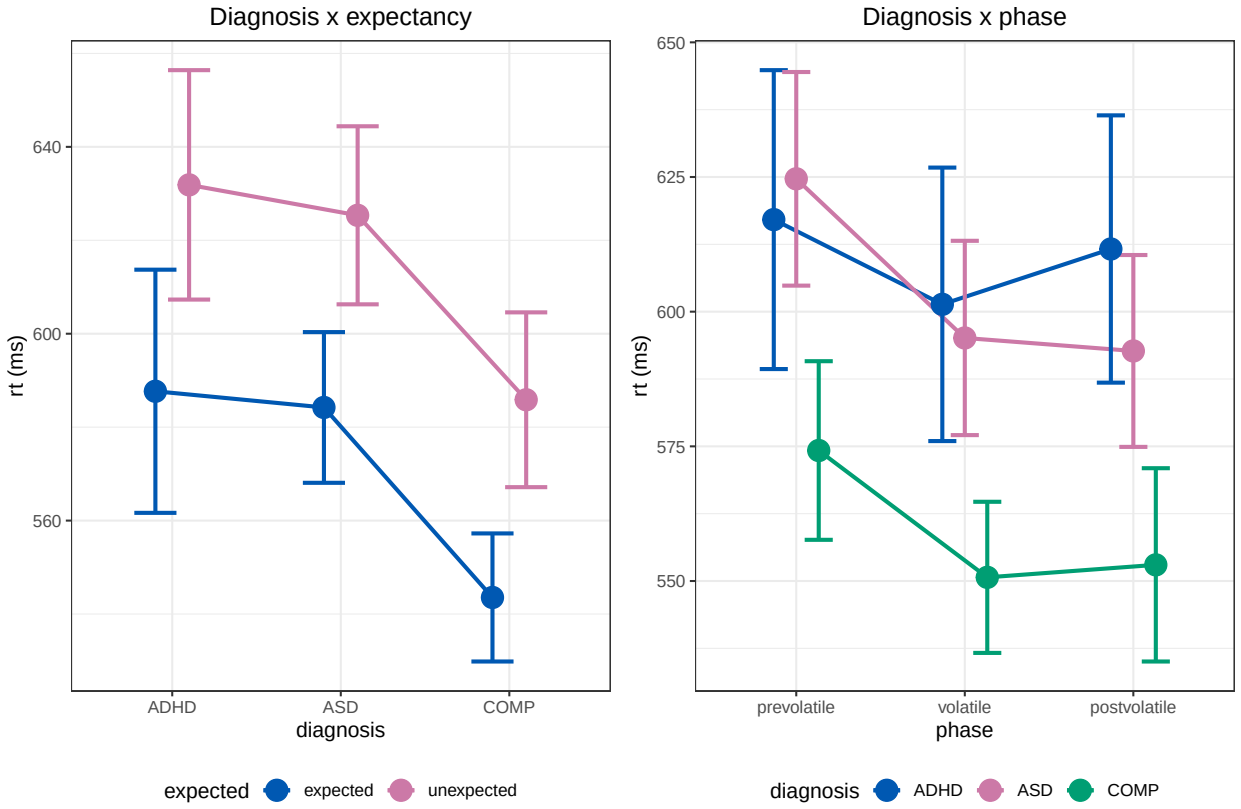
```
# rain cloud plot including all factors
a = 0.66
df.pal %>%
  ggplot(aes(expected, rt.cor, fill = diagnosis, colour = diagnosis)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show.legend = FALSE, alpha = a),
violin.args = list(color = "black", outlier.shape = NA, alpha = a),
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.col) +
  scale_color_manual(values = custom.col) +
  #ylim(0, 1) +
  labs(title = "Reaction times per subject", x = "", y = "rt (ms)") +
  facet_wrap(difficulty ~ phase) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal", text = element_text(size = 15))
```


Reaction times per subject
easy
prevolatile
volatile
postvolatile
medium
difficult
rt (ms)
1250
1000
750
500
250
expected
unexpected
diagnosis
ADHD
ASD
COMP

```r
# rain cloud plot excluding difficulty
a = 0.66
df.pal %>%
  group_by(subID, diagnosis, expected, phase) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  ) %>%
  ggplot(aes(expected, rt.cor, fill = diagnosis, colour = diagnosis)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show.legend = FALSE, alpha = a),
violin.args = list(color = "black", outlier.shape = NA, alpha = a),
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.col) +
  scale_color_manual(values = custom.col) +
  #ylim(0, 1) +
  labs(title = "Reaction times per subject", x = "", y = "rt (ms)") +
  facet_wrap(. ~ phase) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal", text = element_text(size = 15))
```

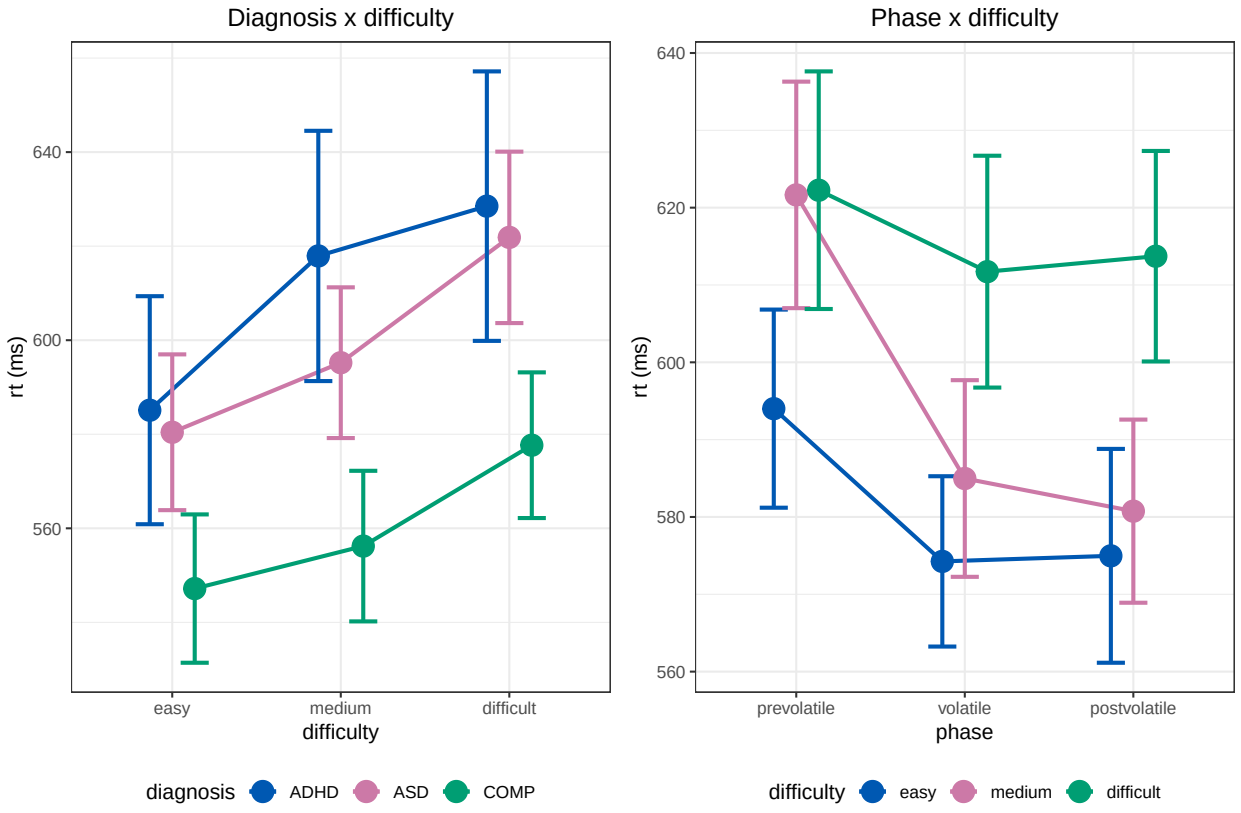


```r
# two-way interactions
p1 = df.pal %>%
  group_by(subID, diagnosis, expected) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  ) %>%
  group_by(diagnosis, expected) %>%
  summarise(
    rt.mn = mean(rt.cor),
    rt.se = sd(rt.cor) / sqrt(n())
  ) %>%
  ggplot(aes(y = rt.mn, x = diagnosis, group = expected, colour = expected)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = rt.mn - rt.se,
                    ymax = rt.mn + rt.se), linewidth = 1, width = 0.5,
                position = position_dodge(0.4)) +
  geom_point(position = position_dodge(0.4), size = 5) +
  labs (x = "diagnosis", y = "rt (ms)") +
  ggtitle("Diagnosis x expectancy") +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))
p2 = df.pal %>%
  group_by(subID, diagnosis, phase) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  )  %>%
  group_by(diagnosis, phase) %>%
  summarise(
    rt.mn = mean(rt.cor),
    rt.se = sd(rt.cor) / sqrt(n())
  ) %>%
  ggplot(aes(y = rt.mn, x = phase, group = diagnosis, colour = diagnosis)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = rt.mn - rt.se,
                    ymax = rt.mn + rt.se), linewidth = 1, width = 0.5,
                position = position_dodge(0.4)) +
  geom_point(position = position_dodge(0.4), size = 5) +
  labs (x = "phase", y = "rt (ms)") +
  ggtitle("Diagnosis x phase") +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))

ggarrange(p1, p2, ncol = 2)
```



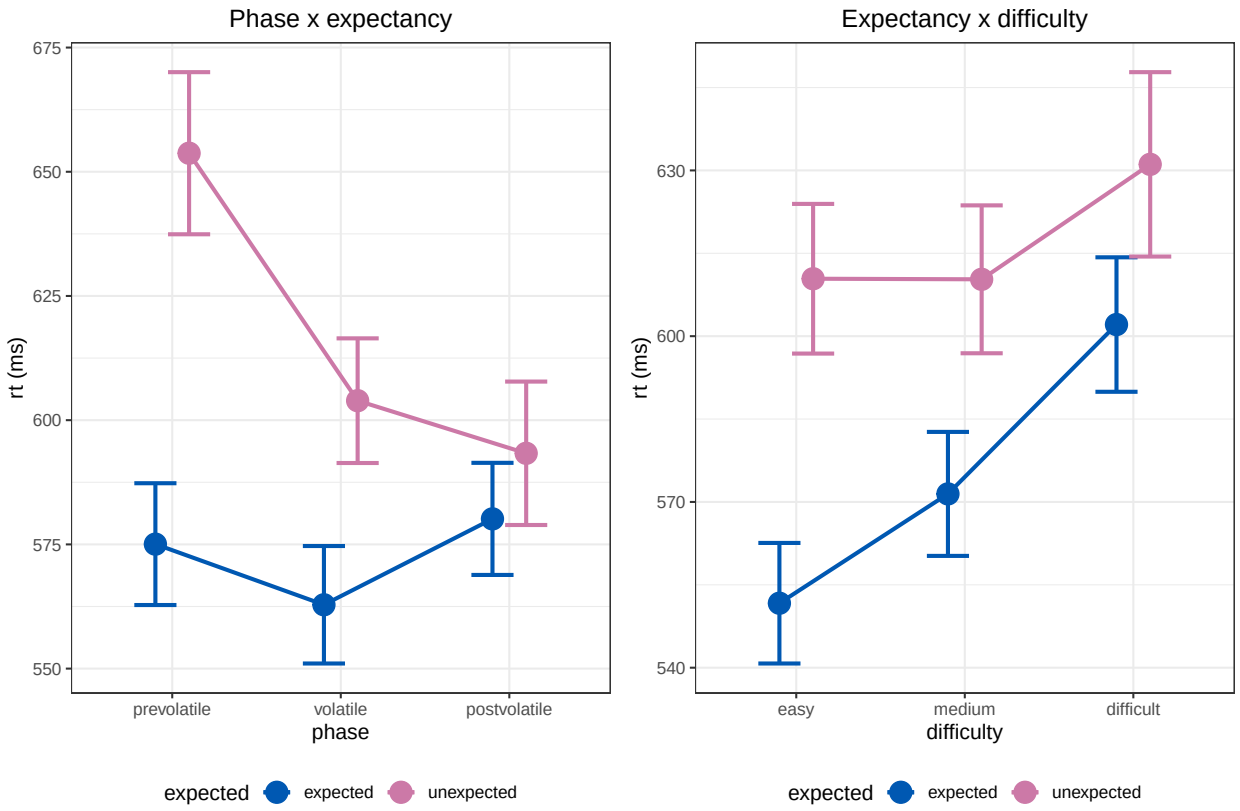

```
p3 = df.pal %>%
  group_by(subID, diagnosis, difficulty) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  )  %>%
  group_by(diagnosis, difficulty) %>%
  summarise(
    rt.mn = mean(rt.cor),
    rt.se = sd(rt.cor) / sqrt(n())
  ) %>%
  ggplot(aes(y = rt.mn, x = difficulty, group = diagnosis, colour = diagnosis)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = rt.mn - rt.se,
                    ymax = rt.mn + rt.se), linewidth = 1, width = 0.5,
                position = position_dodge(0.4)) +
  geom_point(position = position_dodge(0.4), size = 5) +
  labs (x = "difficulty", y = "rt (ms)") +
  ggtitle("Diagnosis x difficulty") +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))
p4 = df.pal %>%
  group_by(subID, phase, difficulty) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  )  %>%
```

```
  group_by(phase, difficulty) %>%
  summarise(
    rt.mn = mean(rt.cor),
    rt.se = sd(rt.cor) / sqrt(n())
  ) %>%
  ggplot(aes(y = rt.mn, x = phase, group = difficulty, colour = difficulty)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = rt.mn - rt.se,
                    ymax = rt.mn + rt.se), linewidth = 1, width = 0.5,
                position = position_dodge(0.4)) +
  geom_point(position = position_dodge(0.4), size = 5) +
  labs (x = "phase", y = "rt (ms)") +
  ggtitle("Phase x difficulty") +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))

ggarrange(p3, p4, ncol = 2)
```

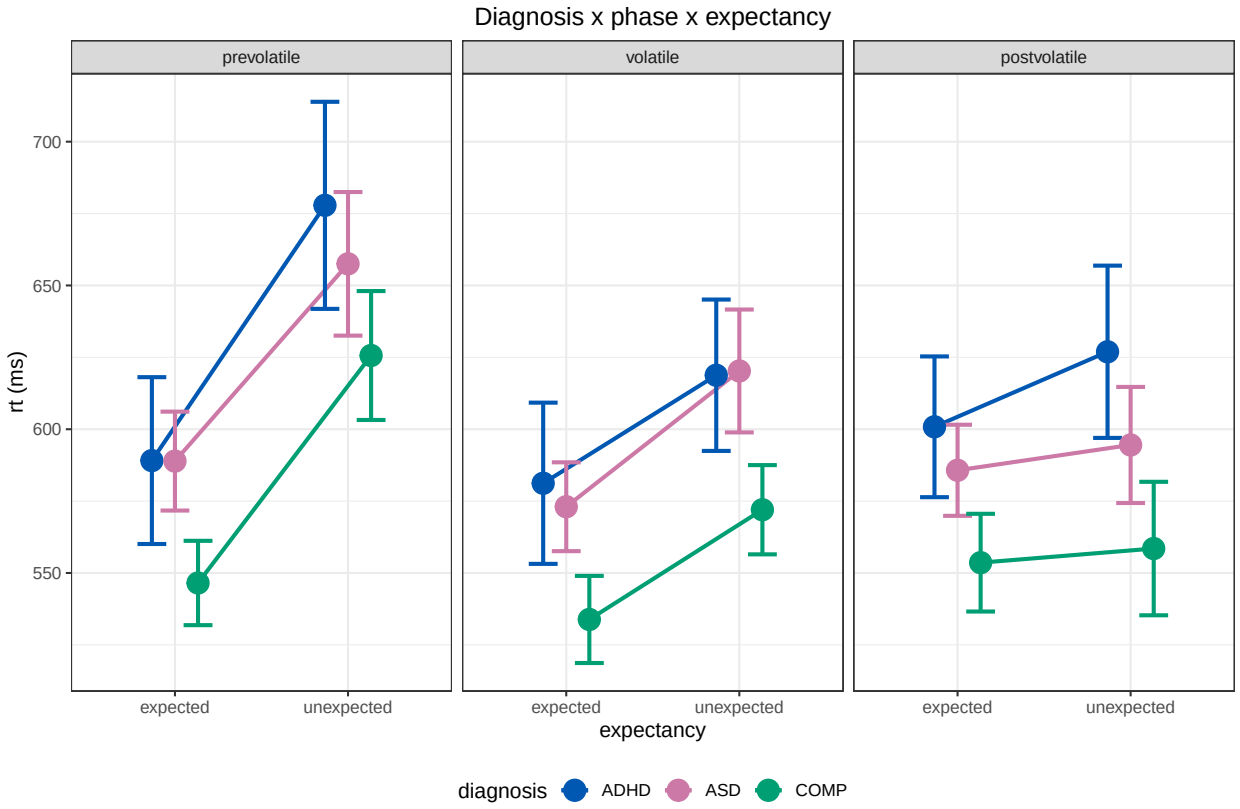



```
p5 = df.pal %>%
  group_by(subID, expected, phase) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  )  %>%
  group_by(phase, expected) %>%
```

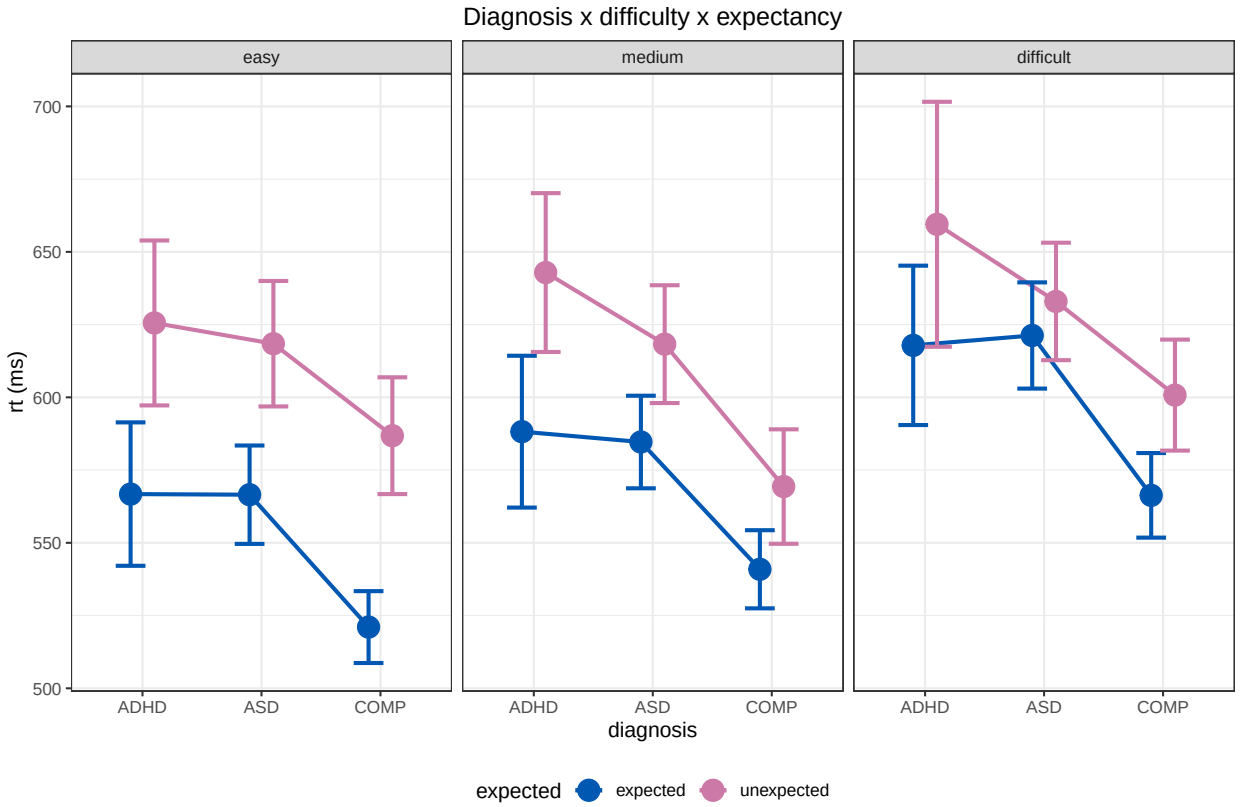
```
  summarise(
    rt.mn = mean(rt.cor),
    rt.se = sd(rt.cor) / sqrt(n())
  ) %>%
  ggplot(aes(y = rt.mn, x = phase, group = expected, colour = expected)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = rt.mn - rt.se,
                    ymax = rt.mn + rt.se), linewidth = 1, width = 0.5,
                position = position_dodge(0.4)) +
  geom_point(position = position_dodge(0.4), size = 5) +
  labs (x = "phase", y = "rt (ms)") +
  ggtitle("Phase x expectancy") +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))
p6 = df.pal %>%
  group_by(subID, expected, difficulty) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  )  %>%
  group_by(expected, difficulty) %>%
  summarise(
    rt.mn = mean(rt.cor),
    rt.se = sd(rt.cor) / sqrt(n())
  ) %>%
  ggplot(aes(y = rt.mn, x = difficulty, group = expected, colour = expected)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = rt.mn - rt.se,
                    ymax = rt.mn + rt.se), linewidth = 1, width = 0.5,
                position = position_dodge(0.4)) +
  geom_point(position = position_dodge(0.4), size = 5) +
  labs (x = "difficulty", y = "rt (ms)") +
  ggtitle("Expectancy x difficulty") +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))

ggarrange(p5, p6, ncol = 2)
```



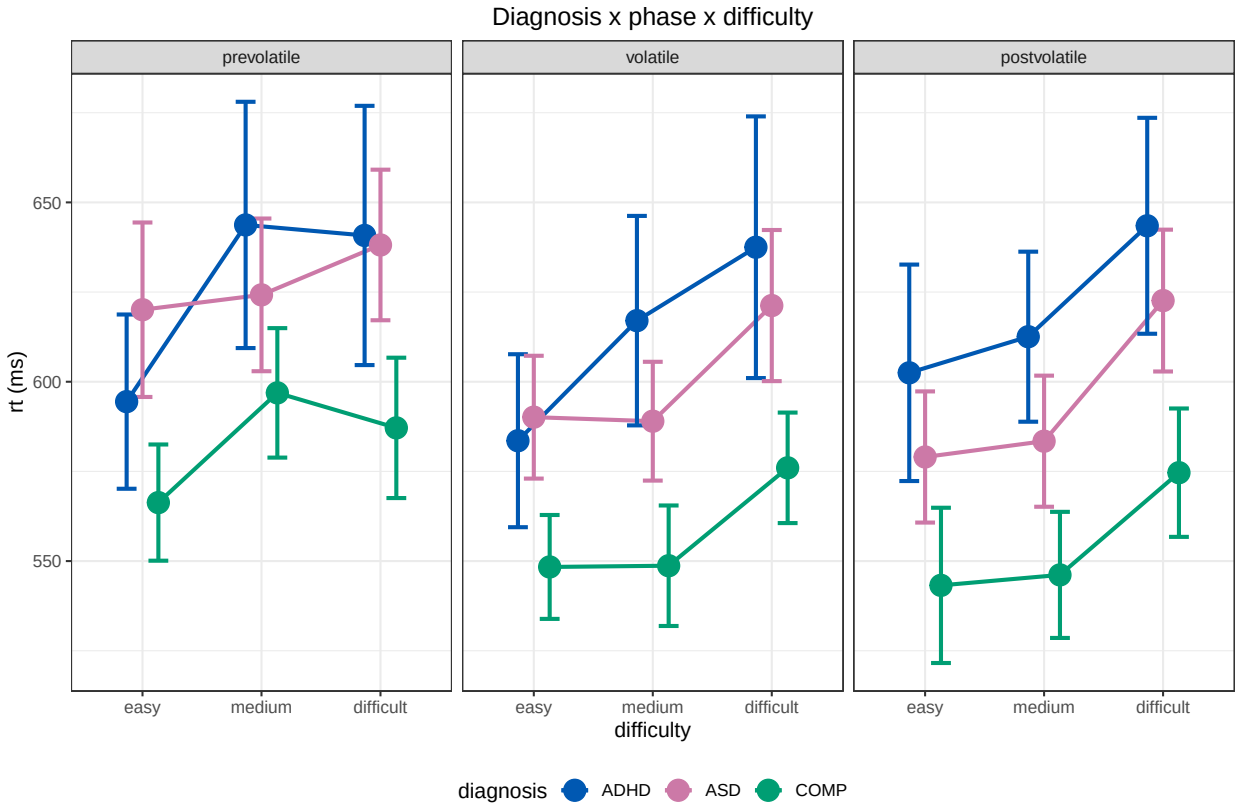

```r
# three-way interaction
df.pal %>%
  group_by(subID, diagnosis, expected, phase) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  )  %>%
  group_by(diagnosis, expected, phase) %>%
  summarise(
    rt.mn = mean(rt.cor),
    rt.se = sd(rt.cor) / sqrt(n())
  ) %>%
  ggplot(aes(y = rt.mn, x = expected, group = diagnosis, colour = diagnosis)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = rt.mn - rt.se,
                    ymax = rt.mn + rt.se), linewidth = 1, width = 0.5,
                position = position_dodge(0.4)) +
  geom_point(position = position_dodge(0.4), size = 5) +
  labs (x = "expectancy", y = "rt (ms)") +
  ggtitle("Diagnosis x phase x expectancy") +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  facet_wrap(. ~ phase) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))
```



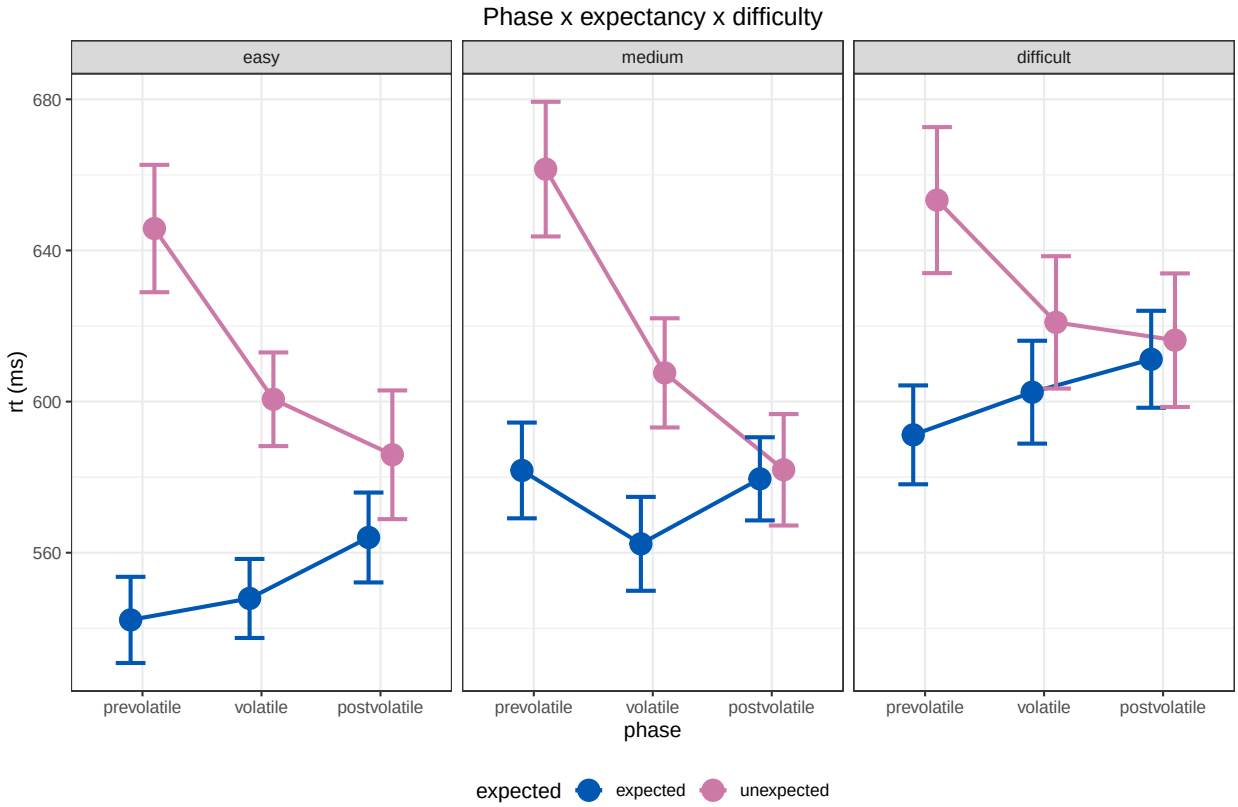

```
df.pal %>%
  group_by(subID, diagnosis, expected, difficulty) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  )  %>%
  group_by(diagnosis, expected, difficulty) %>%
  summarise(
    rt.mn = mean(rt.cor),
    rt.se = sd(rt.cor) / sqrt(n())
  ) %>%
  ggplot(aes(y = rt.mn, x = diagnosis, group = expected, colour = expected)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = rt.mn - rt.se,
                    ymax = rt.mn + rt.se), linewidth = 1, width = 0.5,
                position = position_dodge(0.4)) +
  geom_point(position = position_dodge(0.4), size = 5) +
  labs (x = "diagnosis", y = "rt (ms)") +
  ggtitle("Diagnosis x difficulty x expectancy") +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  facet_wrap(. ~ difficulty) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))
```




```
df.pal %>%
  group_by(subID, diagnosis, phase, difficulty) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  )  %>%
  group_by(diagnosis, difficulty, phase) %>%
  summarise(
    rt.mn = mean(rt.cor),
    rt.se = sd(rt.cor) / sqrt(n())
  ) %>%
  ggplot(aes(y = rt.mn, x = difficulty, group = diagnosis, colour = diagnosis)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = rt.mn - rt.se,
                    ymax = rt.mn + rt.se), linewidth = 1, width = 0.5,
                position = position_dodge(0.4)) +
  geom_point(position = position_dodge(0.4), size = 5) +
  labs (x = "difficulty", y = "rt (ms)") +
  ggtitle("Diagnosis x phase x difficulty") +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  facet_wrap(. ~ phase) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))
```




```
df.pal %>%
  group_by(subID, expected, phase, difficulty) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  )  %>%
  group_by(expected, phase, difficulty) %>%
  summarise(
    rt.mn = mean(rt.cor),
    rt.se = sd(rt.cor) / sqrt(n())
  ) %>%
  ggplot(aes(y = rt.mn, x = phase, group = expected, colour = expected)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = rt.mn - rt.se,
                    ymax = rt.mn + rt.se), linewidth = 1, width = 0.5,
                position = position_dodge(0.4)) +
  geom_point(position = position_dodge(0.4), size = 5) +
  labs (x = "phase", y = "rt (ms)") +
  ggtitle("Phase x expectancy x difficulty") +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  facet_wrap(. ~ difficulty) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))
```

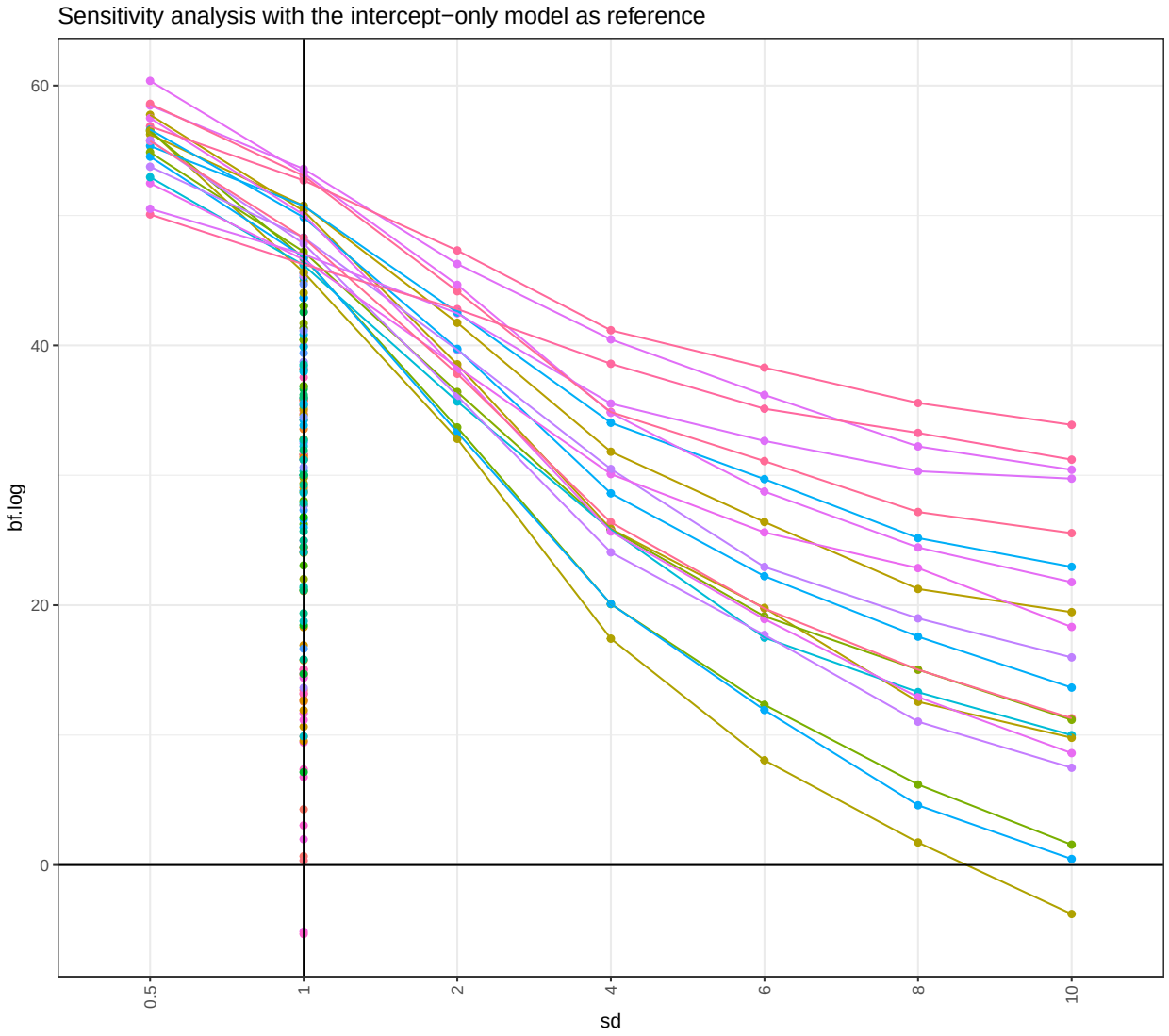


Bayes factor analysis

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

# rerun the model with more iterations for bridgesampling
m.pal.bf = brm(f.pal,
          df.pal, prior = priors,
          family = shifted_lognormal,
          iter = 40000, warmup = 10000,
          backend = "cmdstanr", threads = threading(8),
          file = "m_pal_bf", silent = 2,
          save_pars = save_pars(all = TRUE)
          )

# first, run it only for the chosen parameters to determine the best models
tryCatch({
  # check if it has been run before
  if (!file.exists(file.path(sense_dir,
                             sprintf("df_%s_bf.csv", main.code)))) {
    # use function to compute BF with the described priors
    bf_sens_4int(m.pal.bf, "diagnosis", "expected", "phase", "difficulty",
                 "chosen",
                 main.code, # prefix for all models and MLL
                 file.path("logfiles", "log_PAL_bf.txt"), # log file
```

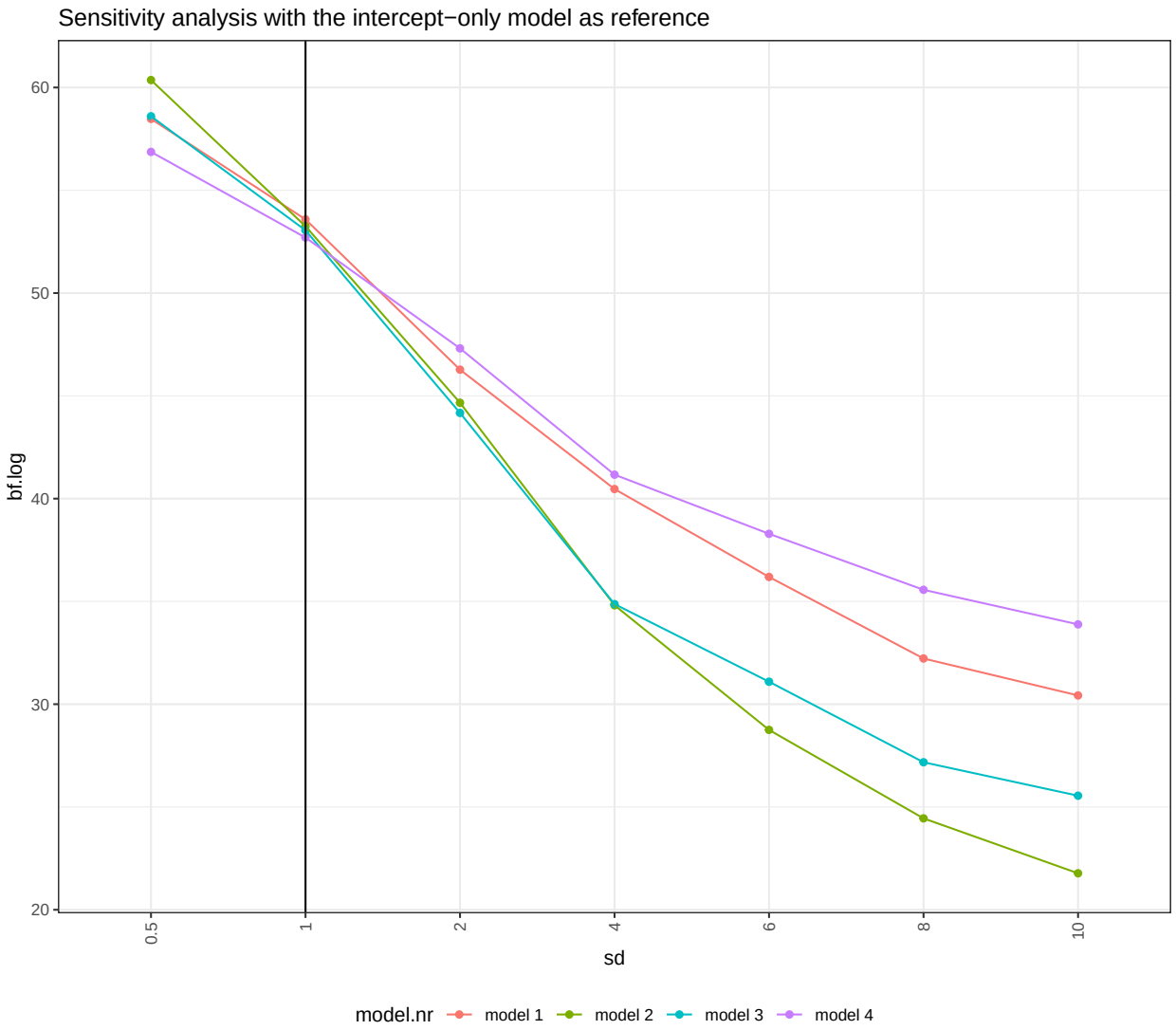
```
                  sense_dir # where to save the models and MLL
    )
  }
  },
  error = function(err) {
    message(sprintf("Error for %s: %s", "chosen", err))
  }
)

# since there are 167 models in total, we focus on the 20 best models for the
# sensitivity analysis
df.pal.bf = read_csv(file.path(sense_dir,
                               sprintf("df_%s_bf.csv", main.code)),
                     show_col_types = F) %>%
  ungroup() %>%
  filter(priors == "chosen") %>%
  mutate(bf.rank = rank(bf.log),
         bf.rank = max(bf.rank) - bf.rank + 1
         ) %>%
  filter(bf.rank <= 20)
fixed = df.pal.bf$`population-level`

# and describe the priors we want to use in our sensitivity analysis
pr.descriptions = c("sdx0.5",
                    "sdx2",    "sdx4",   "sdx6",     "sdx8",     "sdx10"
  )

# check if they have been run already
pr.done = read_csv(file.path(sense_dir,
                               sprintf("df_%s_bf.csv", main.code)),
                     show_col_types = F) %>%
    select(priors) %>% distinct()
pr.descriptions = pr.descriptions[!(pr.descriptions %in% pr.done$priors)]

# loop through the priors that have not been used before
for (pr.desc in pr.descriptions) {
  tryCatch({
    # use function to compute BF with the described priors
    bf_sens_fixed(m.pal.bf, fixed,
                  pr.desc,
                  main.code, # prefix for all models and MLL
                  file.path("logfiles", "log_PAL_bf.txt"), # log file
                  sense_dir # where to save the models and MLL
    )
  },
  error = function(err) {
    message(sprintf("Error for %s: %s", pr.desc, err))
  }
  )
}

df.pal.bf = read_csv(file.path(sense_dir,
                               sprintf("df_%s_bf.csv", main.code)),
```

```
                    show_col_types = F)

# check the sensitivity analysis result per model
df.pal.bf %>%
  filter(`population-level` != "1") %>%
  mutate(
    sd = as.factor(case_when(
      priors == "chosen" ~ "1",
      substr(priors, 1, 3) == "sdx" ~ gsub("sdx", "", priors),
      T ~ priors)
    ),
    order = case_when(
      priors == "chosen" ~ 1,
      substr(priors, 1, 3) == "sdx" ~ as.numeric(gsub("sdx", "", priors)),
      T ~ 999),
    sd = fct_reorder(sd, order)
  ) %>%
  ggplot(aes(y = bf.log,
             x = sd,
             group = `population-level`,
             colour = `population-level`)) +
  geom_point() +
  geom_line() +
  geom_vline(xintercept = "1") +
  geom_hline(yintercept = 0) +
  ggtitle("Sensitivity analysis with the intercept-only model as reference") +
  #scale_colour_manual(values = custom.col) +
  theme_bw() +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```




```
# There seems to be a bit of a change between the best few models from the chosen
# prior to wider priors, let's inspect this more closely
df.pal.bf.best = df.pal.bf %>%
  # choose the best models based on the chosen priors
  filter(priors == "chosen") %>%
  mutate(
    bf.rank = max(rank(bf.log)) - rank(bf.log) + 1
  ) %>%
  filter(bf.rank < 5) %>%
  # add a model number to make plotting easier
  mutate(
    model.nr = paste("model", bf.rank)
  ) %>%
  select(`population-level`, model.nr) %>%
  # merge with the other data again
  merge(., df.pal.bf)

#kable(df.pal.bf.best %>%
```

```
#  select(-bf.rank) %>%
#  pivot_wider(names_from = priors, values_from = bf.log) %>% arrange(desc(chosen)))

df.pal.bf.best %>%
  mutate(
    sd = as.factor(case_when(
      priors == "chosen" ~ "1",
      substr(priors, 1, 3) == "sdx" ~ gsub("sdx", "", priors),
      T ~ priors)
    ),
    order = case_when(
      priors == "chosen" ~ 1,
      substr(priors, 1, 3) == "sdx" ~ as.numeric(gsub("sdx", "", priors)),
      T ~ 999),
    sd = fct_reorder(sd, order)
  ) %>%
  ggplot(aes(y = bf.log,
             x = sd,
             group = model.nr ,
             colour = model.nr)) +
  geom_point() +
  geom_line() +
  geom_vline(xintercept = "1") +
  ggtitle("Sensitivity analysis with the intercept-only model as reference") +
  theme_bw() +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```



S1.4 Exploration of reaction times in all non-outlier trials

Since other studies have also used all non-outlier trials regardless of accuracy, we compute the above described model on these RTs as well.

```
# fit the final model
set.seed(2468)
m.use = brm(rt.use ~ diagnosis * expected * phase * difficulty +
                  (expected + phase + difficulty +
                     expected:phase + difficulty:phase + expected:difficulty | subID),
            df.pal, prior = priors,
            family = shifted_lognormal,
            iter = iter, warmup = warm,
            backend = "cmdstanr", threads = threading(8),
            file = "m_pal_rtuse",
            save_pars = save_pars(all = TRUE)
            )
rstan::check_hmc_diagnostics(m.use$fit)
```

```
##
## Divergences:
## 0 of 18000 iterations ended with a divergence.
##
## Tree depth:
## 0 of 18000 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.use) >= 1.01, na.rm = T)
```

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

```
# check the trace plots
post.draws = as_draws_df(m.use)
mcmc_trace(post.draws, regex_pars = "^b_",
           facet_args = list(ncol = 6)) +
  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.
```


```
# print the model summary
summary(m.use)
```

```
##  Family: shifted_lognormal
##   Links: mu = identity; sigma = identity; ndt = identity
## Formula: rt.use ~ diagnosis * expected * phase * difficulty + (expected + phase + difficulty + expec
##    Data: df.pal (Number of observations: 1206)
##   Draws: 4 chains, each with iter = 6000; warmup = 1500; thin = 1;
##          total post-warmup draws = 18000
##
## Multilevel Hyperparameters:
## ~subID (Number of levels: 67)
##                                                 Estimate Est.Error l-95% CI
## sd(Intercept)                                       0.23      0.02     0.19
```

```
## sd(expected1)                                  0.02     0.01     0.01
## sd(phase1)                                     0.08     0.01     0.06
## sd(phase2)                                     0.02     0.01     0.00
## sd(difficulty1)                                0.01     0.01     0.00
## sd(difficulty2)                                0.01     0.01     0.00
## sd(expected1:phase1)                           0.02     0.01     0.01
## sd(expected1:phase2)                           0.02     0.01     0.00
## sd(phase1:difficulty1)                         0.04     0.01     0.02
## sd(phase2:difficulty1)                         0.02     0.01     0.00
## sd(phase1:difficulty2)                         0.03     0.01     0.00
## sd(phase2:difficulty2)                         0.03     0.01     0.01
## sd(expected1:difficulty1)                      0.01     0.01     0.00
## sd(expected1:difficulty2)                      0.02     0.01     0.00
## cor(Intercept,expected1)                      -0.16     0.16    -0.47
## cor(Intercept,phase1)                          0.03     0.12    -0.19
## cor(expected1,phase1)                          0.26     0.17    -0.10
## cor(Intercept,phase2)                         -0.08     0.18    -0.43
## cor(expected1,phase2)                          0.07     0.22    -0.36
## cor(phase1,phase2)                            -0.05     0.19    -0.41
## cor(Intercept,difficulty1)                    -0.10     0.22    -0.50
## cor(expected1,difficulty1)                    -0.00     0.23    -0.45
## cor(phase1,difficulty1)                       -0.09     0.22    -0.52
## cor(phase2,difficulty1)                       -0.08     0.24    -0.52
## cor(Intercept,difficulty2)                    -0.10     0.20    -0.48
## cor(expected1,difficulty2)                    -0.13     0.22    -0.54
## cor(phase1,difficulty2)                       -0.26     0.21    -0.63
## cor(phase2,difficulty2)                       -0.07     0.23    -0.51
## cor(difficulty1,difficulty2)                  -0.01     0.24    -0.47
## cor(Intercept,expected1:phase1)                0.02     0.17    -0.32
## cor(expected1,expected1:phase1)                0.00     0.21    -0.41
## cor(phase1,expected1:phase1)                  -0.28     0.18    -0.60
## cor(phase2,expected1:phase1)                  -0.11     0.22    -0.52
## cor(difficulty1,expected1:phase1)             -0.01     0.23    -0.46
## cor(difficulty2,expected1:phase1)              0.09     0.22    -0.36
## cor(Intercept,expected1:phase2)                0.14     0.20    -0.29
## cor(expected1,expected1:phase2)               -0.10     0.22    -0.52
## cor(phase1,expected1:phase2)                  -0.13     0.21    -0.52
## cor(phase2,expected1:phase2)                  -0.24     0.24    -0.66
## cor(difficulty1,expected1:phase2)              0.03     0.24    -0.43
## cor(difficulty2,expected1:phase2)              0.10     0.23    -0.36
## cor(expected1:phase1,expected1:phase2)         0.08     0.23    -0.36
## cor(Intercept,phase1:difficulty1)             -0.31     0.15    -0.60
## cor(expected1,phase1:difficulty1)             -0.08     0.21    -0.48
## cor(phase1,phase1:difficulty1)                 0.09     0.17    -0.25
## cor(phase2,phase1:difficulty1)                -0.01     0.21    -0.42
## cor(difficulty1,phase1:difficulty1)            0.06     0.23    -0.40
## cor(difficulty2,phase1:difficulty1)            0.05     0.22    -0.39
## cor(expected1:phase1,phase1:difficulty1)      -0.12     0.21    -0.51
## cor(expected1:phase2,phase1:difficulty1)      -0.10     0.22    -0.53
## cor(Intercept,phase2:difficulty1)             -0.19     0.20    -0.54
## cor(expected1,phase2:difficulty1)             -0.06     0.22    -0.48
## cor(phase1,phase2:difficulty1)                -0.17     0.20    -0.55
## cor(phase2,phase2:difficulty1)                -0.08     0.23    -0.51
## cor(difficulty1,phase2:difficulty1)            0.04     0.24    -0.43
```

```
## cor(difficulty2,phase2:difficulty1)                 0.16      0.23    -0.32
## cor(expected1:phase1,phase2:difficulty1)            0.11      0.23    -0.34
## cor(expected1:phase2,phase2:difficulty1)            0.12      0.23    -0.35
## cor(phase1:difficulty1,phase2:difficulty1)         -0.03      0.22    -0.45
## cor(Intercept,phase1:difficulty2)                   0.12      0.18    -0.25
## cor(expected1,phase1:difficulty2)                   0.13      0.22    -0.31
## cor(phase1,phase1:difficulty2)                      0.03      0.19    -0.34
## cor(phase2,phase1:difficulty2)                      0.15      0.23    -0.31
## cor(difficulty1,phase1:difficulty2)                -0.11      0.24    -0.55
## cor(difficulty2,phase1:difficulty2)                -0.17      0.23    -0.58
## cor(expected1:phase1,phase1:difficulty2)           -0.00      0.22    -0.43
## cor(expected1:phase2,phase1:difficulty2)           -0.11      0.23    -0.53
## cor(phase1:difficulty1,phase1:difficulty2)         -0.17      0.22    -0.56
## cor(phase2:difficulty1,phase1:difficulty2)         -0.14      0.23    -0.56
## cor(Intercept,phase2:difficulty2)                   0.08      0.18    -0.29
## cor(expected1,phase2:difficulty2)                   0.14      0.21    -0.30
## cor(phase1,phase2:difficulty2)                     -0.07      0.19    -0.43
## cor(phase2,phase2:difficulty2)                     -0.02      0.22    -0.45
## cor(difficulty1,phase2:difficulty2)                -0.01      0.23    -0.45
## cor(difficulty2,phase2:difficulty2)                -0.09      0.23    -0.52
## cor(expected1:phase1,phase2:difficulty2)            0.21      0.22    -0.24
## cor(expected1:phase2,phase2:difficulty2)            0.07      0.23    -0.39
## cor(phase1:difficulty1,phase2:difficulty2)         -0.20      0.21    -0.59
## cor(phase2:difficulty1,phase2:difficulty2)         -0.15      0.24    -0.58
## cor(phase1:difficulty2,phase2:difficulty2)          0.09      0.22    -0.36
## cor(Intercept,expected1:difficulty1)                0.05      0.23    -0.42
## cor(expected1,expected1:difficulty1)               -0.00      0.24    -0.46
## cor(phase1,expected1:difficulty1)                   0.03      0.23    -0.43
## cor(phase2,expected1:difficulty1)                  -0.01      0.24    -0.47
## cor(difficulty1,expected1:difficulty1)             -0.05      0.24    -0.51
## cor(difficulty2,expected1:difficulty1)              0.00      0.24    -0.46
## cor(expected1:phase1,expected1:difficulty1)         0.04      0.24    -0.43
## cor(expected1:phase2,expected1:difficulty1)         0.03      0.24    -0.44
## cor(phase1:difficulty1,expected1:difficulty1)       0.03      0.24    -0.44
## cor(phase2:difficulty1,expected1:difficulty1)       0.04      0.24    -0.43
## cor(phase1:difficulty2,expected1:difficulty1)      -0.03      0.24    -0.48
## cor(phase2:difficulty2,expected1:difficulty1)      -0.02      0.24    -0.48
## cor(Intercept,expected1:difficulty2)                0.01      0.19    -0.35
## cor(expected1,expected1:difficulty2)                0.22      0.22    -0.22
## cor(phase1,expected1:difficulty2)                   0.33      0.19    -0.09
## cor(phase2,expected1:difficulty2)                   0.06      0.22    -0.38
## cor(difficulty1,expected1:difficulty2)             -0.03      0.24    -0.48
## cor(difficulty2,expected1:difficulty2)             -0.26      0.24    -0.66
## cor(expected1:phase1,expected1:difficulty2)        -0.08      0.22    -0.49
## cor(expected1:phase2,expected1:difficulty2)        -0.07      0.23    -0.49
## cor(phase1:difficulty1,expected1:difficulty2)      -0.07      0.21    -0.47
## cor(phase2:difficulty1,expected1:difficulty2)      -0.12      0.23    -0.55
## cor(phase1:difficulty2,expected1:difficulty2)       0.13      0.22    -0.33
## cor(phase2:difficulty2,expected1:difficulty2)       0.16      0.22    -0.31
## cor(expected1:difficulty1,expected1:difficulty2)   -0.08      0.25    -0.54
##                                                u-95% CI Rhat Bulk_ESS
## sd(Intercept)                                      0.28 1.00     1842
## sd(expected1)                                      0.03 1.00     3698
## sd(phase1)                                         0.10 1.00     4988
```

```
## sd(phase2)                                        0.04 1.00     2061
## sd(difficulty1)                                   0.03 1.00     3878
## sd(difficulty2)                                   0.03 1.00     4205
## sd(expected1:phase1)                              0.04 1.00     3860
## sd(expected1:phase2)                              0.03 1.00     3494
## sd(phase1:difficulty1)                            0.06 1.00     5142
## sd(phase2:difficulty1)                            0.04 1.00     3322
## sd(phase1:difficulty2)                            0.05 1.00     2837
## sd(phase2:difficulty2)                            0.05 1.00     3468
## sd(expected1:difficulty1)                         0.02 1.00     6924
## sd(expected1:difficulty2)                         0.03 1.00     4479
## cor(Intercept,expected1)                          0.18 1.00    15907
## cor(Intercept,phase1)                             0.25 1.00     6108
## cor(expected1,phase1)                             0.58 1.00     1702
## cor(Intercept,phase2)                             0.29 1.00    14646
## cor(expected1,phase2)                             0.48 1.00     8921
## cor(phase1,phase2)                                0.36 1.00    11275
## cor(Intercept,difficulty1)                        0.34 1.00    16485
## cor(expected1,difficulty1)                        0.45 1.00    14190
## cor(phase1,difficulty1)                           0.36 1.00    15920
## cor(phase2,difficulty1)                           0.39 1.00    11752
## cor(Intercept,difficulty2)                        0.31 1.00    18286
## cor(expected1,difficulty2)                        0.32 1.00    10887
## cor(phase1,difficulty2)                           0.20 1.00    13212
## cor(phase2,difficulty2)                           0.39 1.00    11734
## cor(difficulty1,difficulty2)                      0.45 1.00    11419
## cor(Intercept,expected1:phase1)                   0.35 1.00    14600
## cor(expected1,expected1:phase1)                   0.41 1.00     7709
## cor(phase1,expected1:phase1)                      0.09 1.00    11518
## cor(phase2,expected1:phase1)                      0.32 1.00     8833
## cor(difficulty1,expected1:phase1)                 0.44 1.00     8565
## cor(difficulty2,expected1:phase1)                 0.52 1.00     8195
## cor(Intercept,expected1:phase2)                   0.50 1.00    16011
## cor(expected1,expected1:phase2)                   0.36 1.00    10698
## cor(phase1,expected1:phase2)                      0.31 1.00    12989
## cor(phase2,expected1:phase2)                      0.27 1.00     4931
## cor(difficulty1,expected1:phase2)                 0.48 1.00    12532
## cor(difficulty2,expected1:phase2)                 0.53 1.00    10509
## cor(expected1:phase1,expected1:phase2)            0.51 1.00    14157
## cor(Intercept,phase1:difficulty1)                 0.01 1.00    13562
## cor(expected1,phase1:difficulty1)                 0.33 1.00     7667
## cor(phase1,phase1:difficulty1)                    0.42 1.00    13273
## cor(phase2,phase1:difficulty1)                    0.41 1.00     7908
## cor(difficulty1,phase1:difficulty1)               0.49 1.00     7265
## cor(difficulty2,phase1:difficulty1)               0.48 1.00     8292
## cor(expected1:phase1,phase1:difficulty1)          0.29 1.00     8580
## cor(expected1:phase2,phase1:difficulty1)          0.35 1.00     8320
## cor(Intercept,phase2:difficulty1)                 0.24 1.00    12965
## cor(expected1,phase2:difficulty1)                 0.37 1.00    11163
## cor(phase1,phase2:difficulty1)                    0.25 1.00    13689
## cor(phase2,phase2:difficulty1)                    0.38 1.00    11613
## cor(difficulty1,phase2:difficulty1)               0.49 1.00     9877
## cor(difficulty2,phase2:difficulty1)               0.58 1.00     7551
## cor(expected1:phase1,phase2:difficulty1)          0.53 1.00    12763
```

```
## cor(expected1:phase2,phase2:difficulty1)             0.54 1.00     9864
## cor(phase1:difficulty1,phase2:difficulty1)           0.42 1.00    13217
## cor(Intercept,phase1:difficulty2)                    0.48 1.00    14620
## cor(expected1,phase1:difficulty2)                    0.53 1.00     9718
## cor(phase1,phase1:difficulty2)                       0.40 1.00    14729
## cor(phase2,phase1:difficulty2)                       0.57 1.00     5987
## cor(difficulty1,phase1:difficulty2)                  0.37 1.00     7101
## cor(difficulty2,phase1:difficulty2)                  0.32 1.00     7120
## cor(expected1:phase1,phase1:difficulty2)             0.42 1.00    12775
## cor(expected1:phase2,phase1:difficulty2)             0.35 1.00     8009
## cor(phase1:difficulty1,phase1:difficulty2)           0.28 1.00     9297
## cor(phase2:difficulty1,phase1:difficulty2)           0.32 1.00     8833
## cor(Intercept,phase2:difficulty2)                    0.42 1.00    15120
## cor(expected1,phase2:difficulty2)                    0.53 1.00     8759
## cor(phase1,phase2:difficulty2)                       0.31 1.00    14098
## cor(phase2,phase2:difficulty2)                       0.41 1.00     8295
## cor(difficulty1,phase2:difficulty2)                  0.44 1.00     7223
## cor(difficulty2,phase2:difficulty2)                  0.36 1.00     7898
## cor(expected1:phase1,phase2:difficulty2)             0.60 1.00     8455
## cor(expected1:phase2,phase2:difficulty2)             0.51 1.00     9421
## cor(phase1:difficulty1,phase2:difficulty2)           0.24 1.00    10322
## cor(phase2:difficulty1,phase2:difficulty2)           0.34 1.00     7027
## cor(phase1:difficulty2,phase2:difficulty2)           0.51 1.00    11418
## cor(Intercept,expected1:difficulty1)                 0.49 1.00    21724
## cor(expected1,expected1:difficulty1)                 0.46 1.00    20447
## cor(phase1,expected1:difficulty1)                    0.48 1.00    19381
## cor(phase2,expected1:difficulty1)                    0.46 1.00    17071
## cor(difficulty1,expected1:difficulty1)               0.44 1.00    13421
## cor(difficulty2,expected1:difficulty1)               0.48 1.00    15080
## cor(expected1:phase1,expected1:difficulty1)          0.50 1.00    15544
## cor(expected1:phase2,expected1:difficulty1)          0.48 1.00    14470
## cor(phase1:difficulty1,expected1:difficulty1)        0.48 1.00    16423
## cor(phase2:difficulty1,expected1:difficulty1)        0.51 1.00    13161
## cor(phase1:difficulty2,expected1:difficulty1)        0.44 1.00    14490
## cor(phase2:difficulty2,expected1:difficulty1)        0.46 1.00    15549
## cor(Intercept,expected1:difficulty2)                 0.38 1.00    18130
## cor(expected1,expected1:difficulty2)                 0.61 1.00     6791
## cor(phase1,expected1:difficulty2)                    0.65 1.00    13247
## cor(phase2,expected1:difficulty2)                    0.48 1.00    10702
## cor(difficulty1,expected1:difficulty2)               0.43 1.00     9490
## cor(difficulty2,expected1:difficulty2)               0.24 1.00     6713
## cor(expected1:phase1,expected1:difficulty2)          0.35 1.00    13865
## cor(expected1:phase2,expected1:difficulty2)          0.39 1.00    12878
## cor(phase1:difficulty1,expected1:difficulty2)        0.36 1.00    13889
## cor(phase2:difficulty1,expected1:difficulty2)        0.34 1.00    11012
## cor(phase1:difficulty2,expected1:difficulty2)        0.54 1.00    12453
## cor(phase2:difficulty2,expected1:difficulty2)        0.57 1.00     9766
## cor(expected1:difficulty1,expected1:difficulty2)     0.41 1.00    10321
##                                                  Tail_ESS
## sd(Intercept)                                        3967
## sd(expected1)                                        3019
## sd(phase1)                                          10287
## sd(phase2)                                           2380
## sd(difficulty1)                                      5035
```

```
## sd(difficulty2)                                    4890
## sd(expected1:phase1)                               2594
## sd(expected1:phase2)                               4418
## sd(phase1:difficulty1)                             5114
## sd(phase2:difficulty1)                             3653
## sd(phase1:difficulty2)                             2274
## sd(phase2:difficulty2)                             3236
## sd(expected1:difficulty1)                          7509
## sd(expected1:difficulty2)                          3563
## cor(Intercept,expected1)                          13131
## cor(Intercept,phase1)                              9810
## cor(expected1,phase1)                              2863
## cor(Intercept,phase2)                             11261
## cor(expected1,phase2)                             12023
## cor(phase1,phase2)                                10883
## cor(Intercept,difficulty1)                        12130
## cor(expected1,difficulty1)                        12830
## cor(phase1,difficulty1)                           12371
## cor(phase2,difficulty1)                           12950
## cor(Intercept,difficulty2)                        12219
## cor(expected1,difficulty2)                        12558
## cor(phase1,difficulty2)                           10772
## cor(phase2,difficulty2)                           13109
## cor(difficulty1,difficulty2)                      12874
## cor(Intercept,expected1:phase1)                   11926
## cor(expected1,expected1:phase1)                   11562
## cor(phase1,expected1:phase1)                       9562
## cor(phase2,expected1:phase1)                      11959
## cor(difficulty1,expected1:phase1)                 12832
## cor(difficulty2,expected1:phase1)                 12408
## cor(Intercept,expected1:phase2)                   12206
## cor(expected1,expected1:phase2)                   12483
## cor(phase1,expected1:phase2)                      12040
## cor(phase2,expected1:phase2)                       9016
## cor(difficulty1,expected1:phase2)                 14892
## cor(difficulty2,expected1:phase2)                 12892
## cor(expected1:phase1,expected1:phase2)            14452
## cor(Intercept,phase1:difficulty1)                 12835
## cor(expected1,phase1:difficulty1)                 10227
## cor(phase1,phase1:difficulty1)                    12626
## cor(phase2,phase1:difficulty1)                    11828
## cor(difficulty1,phase1:difficulty1)               10578
## cor(difficulty2,phase1:difficulty1)               11774
## cor(expected1:phase1,phase1:difficulty1)          11955
## cor(expected1:phase2,phase1:difficulty1)          12676
## cor(Intercept,phase2:difficulty1)                 10568
## cor(expected1,phase2:difficulty1)                 11921
## cor(phase1,phase2:difficulty1)                    12798
## cor(phase2,phase2:difficulty1)                    12524
## cor(difficulty1,phase2:difficulty1)               13202
## cor(difficulty2,phase2:difficulty1)               12035
## cor(expected1:phase1,phase2:difficulty1)          13409
## cor(expected1:phase2,phase2:difficulty1)          12784
## cor(phase1:difficulty1,phase2:difficulty1)        13546
```

```
## cor(Intercept,phase1:difficulty2)                    12137
## cor(expected1,phase1:difficulty2)                    11790
## cor(phase1,phase1:difficulty2)                       13447
## cor(phase2,phase1:difficulty2)                        8698
## cor(difficulty1,phase1:difficulty2)                  10239
## cor(difficulty2,phase1:difficulty2)                  10692
## cor(expected1:phase1,phase1:difficulty2)             13584
## cor(expected1:phase2,phase1:difficulty2)             10859
## cor(phase1:difficulty1,phase1:difficulty2)           11732
## cor(phase2:difficulty1,phase1:difficulty2)           12676
## cor(Intercept,phase2:difficulty2)                    12413
## cor(expected1,phase2:difficulty2)                    11289
## cor(phase1,phase2:difficulty2)                       13856
## cor(phase2,phase2:difficulty2)                       10547
## cor(difficulty1,phase2:difficulty2)                  10184
## cor(difficulty2,phase2:difficulty2)                  12051
## cor(expected1:phase1,phase2:difficulty2)             10992
## cor(expected1:phase2,phase2:difficulty2)             13920
## cor(phase1:difficulty1,phase2:difficulty2)           11058
## cor(phase2:difficulty1,phase2:difficulty2)           10998
## cor(phase1:difficulty2,phase2:difficulty2)           14299
## cor(Intercept,expected1:difficulty1)                 13085
## cor(expected1,expected1:difficulty1)                 13111
## cor(phase1,expected1:difficulty1)                    14073
## cor(phase2,expected1:difficulty1)                    14143
## cor(difficulty1,expected1:difficulty1)               13926
## cor(difficulty2,expected1:difficulty1)               14664
## cor(expected1:phase1,expected1:difficulty1)          13502
## cor(expected1:phase2,expected1:difficulty1)          14084
## cor(phase1:difficulty1,expected1:difficulty1)        14635
## cor(phase2:difficulty1,expected1:difficulty1)        15206
## cor(phase1:difficulty2,expected1:difficulty1)        14298
## cor(phase2:difficulty2,expected1:difficulty1)        15782
## cor(Intercept,expected1:difficulty2)                 13568
## cor(expected1,expected1:difficulty2)                 11681
## cor(phase1,expected1:difficulty2)                    10301
## cor(phase2,expected1:difficulty2)                    12038
## cor(difficulty1,expected1:difficulty2)               13389
## cor(difficulty2,expected1:difficulty2)               10413
## cor(expected1:phase1,expected1:difficulty2)          13290
## cor(expected1:phase2,expected1:difficulty2)          13636
## cor(phase1:difficulty1,expected1:difficulty2)        14171
## cor(phase2:difficulty1,expected1:difficulty2)        13199
## cor(phase1:difficulty2,expected1:difficulty2)        14583
## cor(phase2:difficulty2,expected1:difficulty2)        12057
## cor(expected1:difficulty1,expected1:difficulty2)     14408
##
## Regression Coefficients:
##                                      Estimate Est.Error l-95% CI u-95% CI
## Intercept                                5.93      0.05     5.84     6.03
## diagnosis1                               0.02      0.03    -0.03     0.07
## diagnosis2                               0.03      0.03    -0.02     0.09
## expected1                               -0.04      0.00    -0.05    -0.03
## phase1                                   0.03      0.01     0.01     0.05
```

```
## phase2                                       -0.01      0.01    -0.02     0.00
## difficulty1                                  -0.05      0.01    -0.06    -0.04
## difficulty2                                  -0.00      0.01    -0.01     0.01
## diagnosis1:expected1                         -0.00      0.01    -0.01     0.01
## diagnosis2:expected1                          0.00      0.01    -0.01     0.02
## diagnosis1:phase1                            -0.03      0.01    -0.06    -0.00
## diagnosis2:phase1                             0.01      0.01    -0.01     0.04
## diagnosis1:phase2                             0.00      0.01    -0.02     0.02
## diagnosis2:phase2                             0.00      0.01    -0.01     0.02
## expected1:phase1                             -0.04      0.01    -0.05    -0.02
## expected1:phase2                             -0.00      0.01    -0.02     0.01
## diagnosis1:difficulty1                       -0.01      0.01    -0.03     0.00
## diagnosis2:difficulty1                        0.01      0.01    -0.00     0.02
## diagnosis1:difficulty2                        0.01      0.01    -0.00     0.03
## diagnosis2:difficulty2                       -0.01      0.01    -0.03     0.00
## expected1:difficulty1                        -0.03      0.01    -0.04    -0.02
## expected1:difficulty2                         0.00      0.01    -0.01     0.01
## phase1:difficulty1                           -0.01      0.01    -0.03     0.01
## phase2:difficulty1                            0.01      0.01    -0.01     0.02
## phase1:difficulty2                            0.02      0.01     0.01     0.04
## phase2:difficulty2                           -0.01      0.01    -0.02     0.01
## diagnosis1:expected1:phase1                   0.00      0.01    -0.01     0.02
## diagnosis2:expected1:phase1                   0.01      0.01    -0.01     0.02
## diagnosis1:expected1:phase2                  -0.00      0.01    -0.02     0.01
## diagnosis2:expected1:phase2                  -0.00      0.01    -0.02     0.01
## diagnosis1:expected1:difficulty1             -0.00      0.01    -0.02     0.01
## diagnosis2:expected1:difficulty1             -0.00      0.01    -0.02     0.01
## diagnosis1:expected1:difficulty2             -0.00      0.01    -0.02     0.01
## diagnosis2:expected1:difficulty2              0.00      0.01    -0.01     0.02
## diagnosis1:phase1:difficulty1                -0.02      0.01    -0.04     0.01
## diagnosis2:phase1:difficulty1                 0.02      0.01    -0.01     0.04
## diagnosis1:phase2:difficulty1                -0.00      0.01    -0.02     0.02
## diagnosis2:phase2:difficulty1                -0.00      0.01    -0.02     0.02
## diagnosis1:phase1:difficulty2                -0.00      0.01    -0.02     0.02
## diagnosis2:phase1:difficulty2                -0.01      0.01    -0.03     0.01
## diagnosis1:phase2:difficulty2                 0.01      0.01    -0.01     0.03
## diagnosis2:phase2:difficulty2                 0.01      0.01    -0.01     0.03
## expected1:phase1:difficulty1                 -0.01      0.01    -0.02     0.01
## expected1:phase2:difficulty1                  0.00      0.01    -0.01     0.02
## expected1:phase1:difficulty2                  0.00      0.01    -0.01     0.01
## expected1:phase2:difficulty2                 -0.01      0.01    -0.03     0.00
## diagnosis1:expected1:phase1:difficulty1       0.00      0.01    -0.01     0.02
## diagnosis2:expected1:phase1:difficulty1       0.01      0.01    -0.01     0.03
## diagnosis1:expected1:phase2:difficulty1       0.00      0.01    -0.02     0.02
## diagnosis2:expected1:phase2:difficulty1       0.00      0.01    -0.02     0.02
## diagnosis1:expected1:phase1:difficulty2       0.00      0.01    -0.02     0.02
## diagnosis2:expected1:phase1:difficulty2      -0.00      0.01    -0.02     0.01
## diagnosis1:expected1:phase2:difficulty2      -0.01      0.01    -0.03     0.01
## diagnosis2:expected1:phase2:difficulty2      -0.01      0.01    -0.03     0.01
##                                        Rhat Bulk_ESS Tail_ESS
## Intercept                              1.00     1971     4045
## diagnosis1                             1.00     1445     3200
## diagnosis2                             1.00     1329     3362
## expected1                              1.00    11558    12995
```

```
## phase1                                    1.00     5116     7888
## phase2                                    1.00    15906    13348
## difficulty1                               1.00    13729    13561
## difficulty2                               1.00    16917    14126
## diagnosis1:expected1                      1.00    11869    13203
## diagnosis2:expected1                      1.00    10732    12041
## diagnosis1:phase1                         1.00     4640     7781
## diagnosis2:phase1                         1.00     4980     8466
## diagnosis1:phase2                         1.00    12602    12482
## diagnosis2:phase2                         1.00    12716    13143
## expected1:phase1                          1.00    12640    12256
## expected1:phase2                          1.00    18196    14284
## diagnosis1:difficulty1                    1.00    13339    13838
## diagnosis2:difficulty1                    1.00    13689    13359
## diagnosis1:difficulty2                    1.00    13209    13045
## diagnosis2:difficulty2                    1.00    13202    13801
## expected1:difficulty1                     1.00    16551    13323
## expected1:difficulty2                     1.00    15677    13109
## phase1:difficulty1                        1.00     9550    12848
## phase2:difficulty1                        1.00    13205    12875
## phase1:difficulty2                        1.00    12673    12874
## phase2:difficulty2                        1.00    12381    13037
## diagnosis1:expected1:phase1               1.00    11950    13376
## diagnosis2:expected1:phase1               1.00    11556    11316
## diagnosis1:expected1:phase2               1.00    14473    14249
## diagnosis2:expected1:phase2               1.00    14282    13285
## diagnosis1:expected1:difficulty1          1.00    13782    13530
## diagnosis2:expected1:difficulty1          1.00    13640    13801
## diagnosis1:expected1:difficulty2          1.00    11930    13339
## diagnosis2:expected1:difficulty2          1.00    12040    12887
## diagnosis1:phase1:difficulty1             1.00    10802    13298
## diagnosis2:phase1:difficulty1             1.00    10249    12100
## diagnosis1:phase2:difficulty1             1.00    11901    13540
## diagnosis2:phase2:difficulty1             1.00    11585    12570
## diagnosis1:phase1:difficulty2             1.00    12316    13252
## diagnosis2:phase1:difficulty2             1.00    11391    12254
## diagnosis1:phase2:difficulty2             1.00    11937    12903
## diagnosis2:phase2:difficulty2             1.00    11350    11676
## expected1:phase1:difficulty1              1.00    14710    13819
## expected1:phase2:difficulty1              1.00    14309    13751
## expected1:phase1:difficulty2              1.00    14797    13651
## expected1:phase2:difficulty2              1.00    14408    14062
## diagnosis1:expected1:phase1:difficulty1 1.00    12245    12874
## diagnosis2:expected1:phase1:difficulty1 1.00    11926    13270
## diagnosis1:expected1:phase2:difficulty1 1.00    12459    13568
## diagnosis2:expected1:phase2:difficulty1 1.00    12293    13229
## diagnosis1:expected1:phase1:difficulty2 1.00    11769    13122
## diagnosis2:expected1:phase1:difficulty2 1.00    12209    12995
## diagnosis1:expected1:phase2:difficulty2 1.00    11806    13163
## diagnosis2:expected1:phase2:difficulty2 1.00    12009    12558
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.12      0.01     0.11     0.14 1.00     3773     7306
```

```
## ndt     202.78     14.54   171.59   228.63 1.00     5111     6733
##
## 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).
```

```
# compare the fixed effects between both models
effects = as.data.frame(fixef(m.pal)) %>%
  mutate(model = "cor") %>%
  rownames_to_column(., var = "fixed effect") %>%
  merge(., as.data.frame(fixef(m.use)) %>%
  mutate(model = "use") %>%
  rownames_to_column(., var = "fixed effect"),
        all = T) %>%
  select(model, `fixed effect`, Estimate) %>%
  pivot_wider(names_from = model, values_from = c(Estimate)) %>%
  mutate(
    Estimate.diff = cor - use
  )

kable(effects)
```

fixed effect	use	cor	Estimate.diff
diagnosis1	0.0206996	0.0242308	0.0035312
diagnosis1:difficulty1	-0.0108713	-0.0176198	-0.0067485
diagnosis1:difficulty2	0.0141523	0.0164241	0.0022718
diagnosis1:expected1	-0.0018535	-0.0090408	-0.0071873
diagnosis1:expected1:difficulty1	-0.0026480	0.0045630	0.0072110
diagnosis1:expected1:difficulty2	-0.0024693	-0.0043975	-0.0019282
diagnosis1:expected1:phase1	0.0032982	0.0017506	-0.0015477
diagnosis1:expected1:phase1:difficulty1	0.0040907	-0.0043766	-0.0084673
diagnosis1:expected1:phase1:difficulty2	0.0029112	0.0043754	0.0014642
diagnosis1:expected1:phase2	-0.0028703	0.0014452	0.0043154
diagnosis1:expected1:phase2:difficulty1	0.0001625	0.0044050	0.0042425
diagnosis1:expected1:phase2:difficulty2	-0.0091272	-0.0106533	-0.0015261
diagnosis1:phase1	-0.0277437	-0.0228245	0.0049191
diagnosis1:phase1:difficulty1	-0.0158611	-0.0039381	0.0119231
diagnosis1:phase1:difficulty2	-0.0026061	-0.0030217	-0.0004156
diagnosis1:phase2	0.0000119	-0.0011395	-0.0011514
diagnosis1:phase2:difficulty1	-0.0013689	-0.0060362	-0.0046673
diagnosis1:phase2:difficulty2	0.0083637	0.0054860	-0.0028777
diagnosis2	0.0344432	0.0330829	-0.0013603
diagnosis2:difficulty1	0.0103908	0.0129049	0.0025142
diagnosis2:difficulty2	-0.0138458	-0.0168930	-0.0030472
diagnosis2:expected1	0.0036899	0.0081956	0.0045057
diagnosis2:expected1:difficulty1	-0.0024001	-0.0058980	-0.0034979
diagnosis2:expected1:difficulty2	0.0045086	0.0066144	0.0021058
diagnosis2:expected1:phase1	0.0075921	0.0085813	0.0009893
diagnosis2:expected1:phase1:difficulty1	0.0069937	0.0128183	0.0058246
diagnosis2:expected1:phase1:difficulty2	-0.0038738	-0.0069224	-0.0030487
diagnosis2:expected1:phase2	-0.0005989	-0.0030904	-0.0024915
diagnosis2:expected1:phase2:difficulty1	0.0023150	-0.0020916	-0.0044065
diagnosis2:expected1:phase2:difficulty2	-0.0087302	-0.0034172	0.0053129
diagnosis2:phase1	0.0136972	0.0098442	-0.0038530

fixed effect	use	cor	Estimate.diff
diagnosis2:phase1:difficulty1	0.0158251	0.0070032	-0.0088219
diagnosis2:phase1:difficulty2	-0.0098921	-0.0083763	0.0015157
diagnosis2:phase2	0.0008858	0.0010766	0.0001908
diagnosis2:phase2:difficulty1	-0.0022244	-0.0004848	0.0017395
diagnosis2:phase2:difficulty2	0.0088506	0.0065845	-0.0022661
difficulty1	-0.0459773	-0.0432881	0.0026893
difficulty2	-0.0012485	-0.0033257	-0.0020772
expected1	-0.0421816	-0.0487712	-0.0065895
expected1:difficulty1	-0.0256102	-0.0237218	0.0018883
expected1:difficulty2	0.0001257	0.0018021	0.0016765
expected1:phase1	-0.0354275	-0.0459287	-0.0105011
expected1:phase1:difficulty1	-0.0080821	-0.0098825	-0.0018004
expected1:phase1:difficulty2	0.0005309	0.0021882	0.0016573
expected1:phase2	-0.0044580	-0.0002871	0.0041709
expected1:phase2:difficulty1	0.0023058	0.0002004	-0.0021054
expected1:phase2:difficulty2	-0.0115939	-0.0106507	0.0009431
Intercept	5.9324553	5.9243231	-0.0081322
phase1	0.0285414	0.0312586	0.0027172
phase1:difficulty1	-0.0102141	-0.0052441	0.0049700
phase1:difficulty2	0.0228001	0.0273711	0.0045710
phase2	-0.0109003	-0.0126631	-0.0017628
phase2:difficulty1	0.0066872	0.0054617	-0.0012255
phase2:difficulty2	-0.0075435	-0.0105323	-0.0029888

S1.4 Exploration of association type

Now, we explore if there were any differences between the tone-valence associations at the beginning.

```
# check normal distribution
df.tp %>% group_by(diagnosis, tone_pic, expected) %>%
  shapiro_test(rt.cor) %>%
    mutate(
      sig = if_else(p < 0.05, "*", "")
    )
```

```
## # A tibble: 12 x 7
##    diagnosis tone_pic       expected   variable statistic        p sig
##    <fct>     <chr>          <fct>      <chr>        <dbl>    <dbl> <chr>
##  1 ADHD      highneg_lowpos expected   rt.cor       0.909 0.0452   "*"
##  2 ADHD      highneg_lowpos unexpected rt.cor       0.811 0.000754 "*"
##  3 ADHD      highpos_lowneg expected   rt.cor       0.874 0.00936  "*"
##  4 ADHD      highpos_lowneg unexpected rt.cor       0.905 0.0372   "*"
##  5 ASD       highneg_lowpos expected   rt.cor       0.883 0.0114   "*"
##  6 ASD       highneg_lowpos unexpected rt.cor       0.945 0.235    ""
##  7 ASD       highpos_lowneg expected   rt.cor       0.959 0.443    ""
##  8 ASD       highpos_lowneg unexpected rt.cor       0.886 0.0134   "*"
##  9 COMP      highneg_lowpos expected   rt.cor       0.928 0.112    ""
## 10 COMP      highneg_lowpos unexpected rt.cor       0.872 0.00861  "*"
## 11 COMP      highpos_lowneg expected   rt.cor       0.918 0.0678   ""
## 12 COMP      highpos_lowneg unexpected rt.cor       0.950 0.323    ""
```

```
# rank transform the data
df.tp = df.tp %>%
  ungroup() %>%
  mutate(
    rrt.cor   = rank(rt.cor),
    tone_pic  = as.factor(tone_pic)
    )

# run the ANOVA
aov.tp = anovaBF(rrt.cor   ~ diagnosis * tone_pic * expected, data = df.tp)
bf.tp  = extractBF(aov.tp, logbf = T)
kable(head(bf.tp %>% arrange(desc(bf.tp))))
```

	bf	error	time	code
diagnosis + tone_pic + expected	6.354197	0.0162941	Fri Jan 31 10:38:24 2025	841e7e0a2b25
diagnosis + tone_pic + expected + tone_pic:expected	5.462159	0.0328489	Fri Jan 31 10:38:24 2025	841e2850ddad
tone_pic + expected	5.056154	0.0085593	Fri Jan 31 10:38:24 2025	841e42f6924c
diagnosis + expected	4.434320	0.0426569	Fri Jan 31 10:38:24 2025	841e4134c9b7
tone_pic + expected + tone_pic:expected	4.095316	0.0121822	Fri Jan 31 10:38:24 2025	841e1275f421
diagnosis + tone_pic + diagnosis:tone_pic + expected	3.996912	0.0290320	Fri Jan 31 10:38:24 2025	841e5b7fe68e

The best model contains all three main effects suggesting that while certain combinations were associated with faster and others with slower reaction times, this was independent of the diagnostic status and expectancy.

S1.5 Explorative analysis of errors

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

```
# check normal distribution
df.pal %>% group_by(diagnosis, phase, expected, difficulty) %>%
  shapiro_test(acc) %>%
    mutate(
      sig = if_else(p < 0.05, "*", "")
    )
```

```
## # A tibble: 54 x 8
##    diagnosis phase       expected   difficulty variable statistic        p sig
##    <fct>     <fct>       <fct>      <fct>      <chr>        <dbl>    <dbl> <chr>
##  1 ADHD      prevolatile expected   easy       acc          0.753  1.00e-4 *
##  2 ADHD      prevolatile expected   medium     acc          0.707  2.40e-5 *
##  3 ADHD      prevolatile expected   difficult  acc          0.848  3.11e-3 *
##  4 ADHD      prevolatile unexpected easy       acc          0.768  1.67e-4 *
##  5 ADHD      prevolatile unexpected medium     acc          0.603  1.43e-6 *
##  6 ADHD      prevolatile unexpected difficult  acc          0.642  3.89e-6 *
##  7 ADHD      volatile    expected   easy       acc          0.794  3.99e-4 *
```

```
##  8 ADHD      volatile    expected   medium     acc          0.836  1.90e-3 *
##  9 ADHD      volatile    expected   difficult  acc          0.837  1.99e-3 *
## 10 ADHD      volatile    unexpected easy       acc          0.756  1.10e-4 *
## # i 44 more rows
```

```
# rank transform the data
df.acc = df.pal %>%
  ungroup() %>%
  mutate(
    racc    = rank(acc)
    )

# run the ANOVA
aov.acc    = anovaBF(racc    ~ diagnosis * phase * expected * difficulty, data = df.acc)
bf.acc = extractBF(aov.acc, logbf = T)
kable(head(bf.acc %>% arrange(desc(bf))))
```

	bf	error	time	code
diagnosis + expected + difficulty	43.53159	0.0146879	Fri Jan 31 10:38:27 2025	841e23a25d20
diagnosis + difficulty	42.68904	0.0096788	Fri Jan 31 10:38:27 2025	841e507b80cb
diagnosis + expected + difficulty + diagnosis:difficulty	42.29085	0.0531848	Fri Jan 31 10:38:30 2025	841e5d80bf8d
diagnosis + difficulty + diagnosis:difficulty	41.39701	0.0112858	Fri Jan 31 10:38:29 2025	841e492bd8ad
diagnosis + expected + difficulty + expected:difficulty	41.06355	0.0230097	Fri Jan 31 10:38:42 2025	841e4d4e445f
diagnosis + expected + diagnosis:expected + difficulty	40.77710	0.0272450	Fri Jan 31 10:38:28 2025	841e2a63f1aa

```
# print overall accuracy rates for all the effects included in the best model
df.acc.diagnosis = df.acc %>%
  group_by(diagnosis) %>%
  summarise(mean_accuracy = mean(acc, na.rm = T),
            sd_accuracy = sd(acc, na.rm = T))
df.acc.diagnosis
```

```
## # A tibble: 3 x 3
##   diagnosis mean_accuracy sd_accuracy
##   <fct>              <dbl>       <dbl>
## 1 ADHD                90.2        12.8
## 2 ASD                 88.8        14.1
## 3 COMP                92.9        11.1
```

```
df.acc %>%
  group_by(expected) %>%
  summarise(mean_accuracy = mean(acc, na.rm = T),
            sd_accuracy = sd(acc, na.rm = T))
```

```
## # A tibble: 2 x 3
##   expected   mean_accuracy sd_accuracy
##   <fct>              <dbl>       <dbl>
## 1 expected            92.2        8.91
```

```
## 2 unexpected          89.1       15.7
```

```
df.acc %>%
  group_by(difficulty) %>%
  summarise(mean_accuracy = mean(acc, na.rm = T),
            sd_accuracy = sd(acc, na.rm = T))
```

```
## # A tibble: 3 x 3
##   difficulty mean_accuracy sd_accuracy
##   <fct>              <dbl>       <dbl>
## 1 easy                92.9        12.1
## 2 medium              92.4        11.7
## 3 difficult           86.5        13.7
```

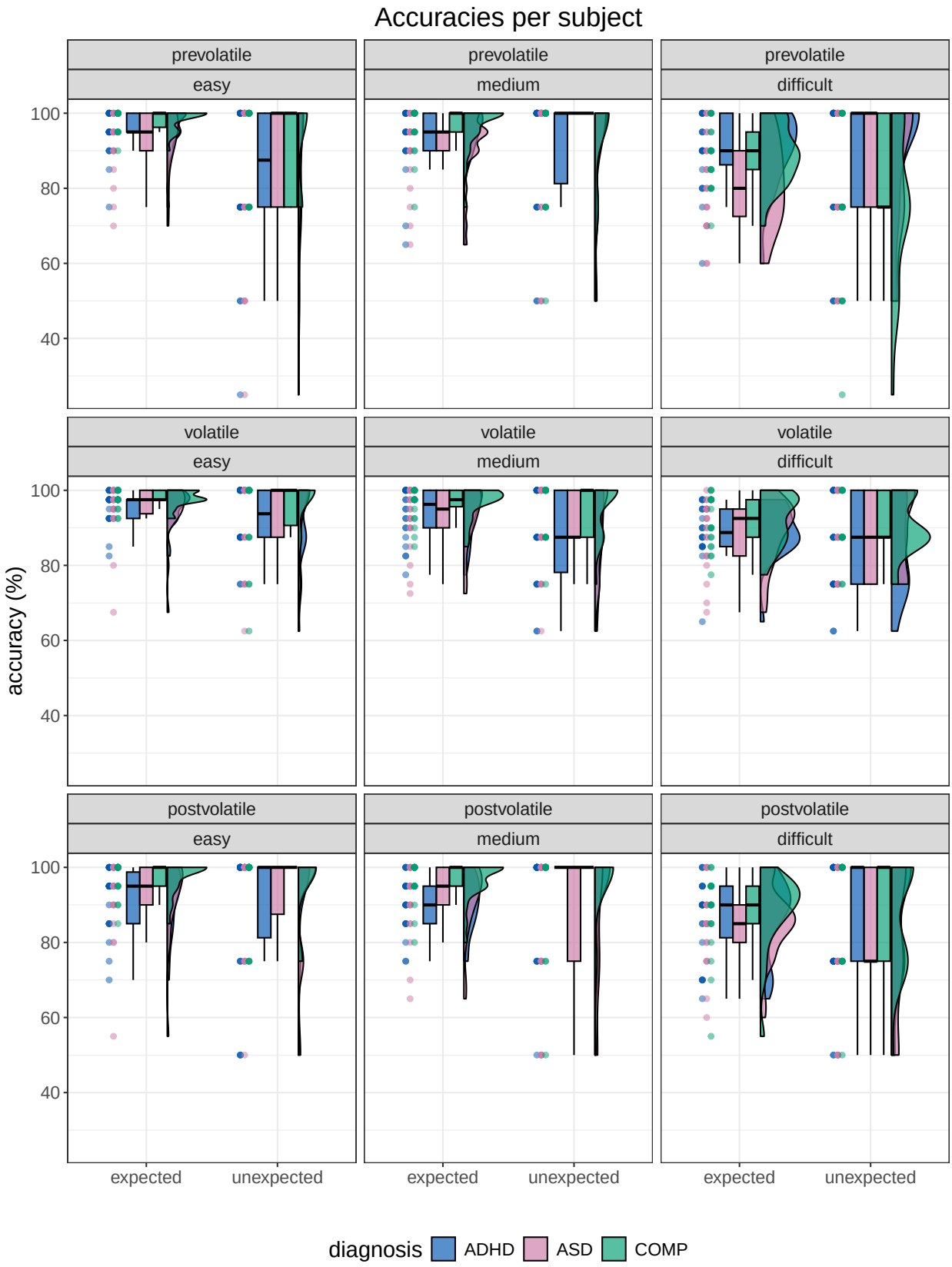
```
df.acc %>%
  group_by(expected, difficulty) %>%
  summarise(mean_accuracy = mean(acc, na.rm = T),
            sd_accuracy = sd(acc, na.rm = T))
```

```
## # A tibble: 6 x 4
## # Groups:   expected [2]
##   expected   difficulty mean_accuracy sd_accuracy
##   <fct>      <fct>              <dbl>       <dbl>
## 1 expected   easy                95.1        6.96
## 2 expected   medium              93.6        7.91
## 3 expected   difficult           87.8        9.91
## 4 unexpected easy                90.7       15.3
## 5 unexpected medium              91.2       14.5
## 6 unexpected difficult           85.3       16.6
```

Accuracies were generally high, with a grand average of 90.64% accurate responses across diagnostic groups.

Plots

```
# rain cloud plot including all factors
a = 0.66
df.acc %>%
  ggplot(aes(expected, acc, fill = diagnosis, colour = diagnosis)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show.legend = FALSE, alpha = a),
violin.args = list(color = "black", outlier.shape = NA, alpha = a),
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.col) +
  scale_color_manual(values = custom.col) +
  labs(title = "Accuracies per subject", x = "", y = "accuracy (%)") +
  facet_wrap(phase ~ difficulty) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal", text = element_text(size = 15))
```


Accuracies per subject
prevolatile
easy
prevolatile
medium
prevolatile
difficult
volatile
easy
volatile
medium
volatile
difficult
postvolatile
easy
postvolatile
medium
postvolatile
difficult
accuracy (%)
100
80
60
40
expected
unexpected
diagnosis
ADHD
ASD
COMP

```r
# plot the two-way interaction
df.acc %>%
  group_by(subID, expected, difficulty) %>%
  summarise(
    acc = mean(acc, na.rm = T)
  ) %>%
  group_by(expected, difficulty) %>%
  summarise(
    acc.mn = mean(acc),
    acc.se = sd(acc) / sqrt(n())
  ) %>%
  ggplot(aes(y = acc.mn, x = difficulty, group = expected, colour = expected)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = acc.mn - acc.se,
                    ymax = acc.mn + acc.se), linewidth = 1, width = 0.5,
                position = position_dodge(0.4)) +
  geom_point(position = position_dodge(0.4), size = 5) +
  labs (x = "difficulty", y = "rt (ms)") +
  ggtitle("Difficulty x expectancy") +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))
```