S1: behavioural, model-agnostic analysis with brms

I. S. Plank

2025-02-26

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] "R.matlab version 3.7.0"
## [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("data/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 = as.factor(recode(diagnosis, "CTR" = "COMP"))
    )

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

  # load the preprocessed data: df.tsk
  df.tsk = rbind(readRDS(file.path(dt.path,  "PAL_tsk.rds")),
                 readRDS(file.path(dt.explo, "PAL_tsk.rds")))

  # load excluded participants (accuracy < 2/3)
  exc = c(scan(file.path(dt.path, 'PAL_exc.txt'),  what="character", sep=NULL),
          scan(file.path(dt.explo, '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)

  # load data and aggregate emotion discrimination threshold
  df.fer = rbind(readRDS(file = paste0(dt.path, '/df_FER.RDS')),
                 readRDS(file = paste0(dt.explo, '/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)

# save the data in the EMBA_HGF toolbox
df.tsk %>% select(subID, diagnosis, ut, difficulty, acc, rt) %>%
  ungroup() %>%
  write_csv(., file = "data/PAL_data.csv")

# anonymise other data in the same way [!MISSING]
#df.et$subID = str_replace_all(df.var$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")
tb.gen = xtabs(~ gender + diagnosis + cis, data = df.sub)

# get the gender descriptions of the not-male and not-female participants
gen.desc = unique(tolower(df.sub[df.sub$gender == "dan",]$gender_desc))

# check which outcomes of interest are normally distributed
df.sht = df.sub %>%
  group_by(diagnosis) %>%
  shapiro_test(age, iq, BDI_total, ASRS_total, RAADS_total, TAS_total) %>%
```

```
  mutate(
    sig = if_else(p < 0.05, "*", "")
  ) %>% arrange(variable)

# most of the measures are not normally distributed;
# therefore, we compute ranks for these outcomes
df.sub = df.sub %>%
  ungroup() %>%
  mutate(
    rASRS  = rank(ASRS_total),
    rage   = rank(age),
    rBDI   = rank(BDI_total),
    rRAADS = rank(RAADS_total),
    rTAS   = rank(TAS_total),
    diagnosis = as.factor(diagnosis)
  )

# now we can compute our ANOVAs
aov.age   = anovaBF(rage   ~ diagnosis, data = df.sub)
aov.iq    = anovaBF(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",])))
`ADHD+ASD`     = sprintf("%.2f (±%.2f)",
                   mean(df.sub[df.sub$diagnosis == "BOTH",]$age),
                   sd(df.sub[df.sub$diagnosis == "BOTH",]$age)/
                     sqrt(nrow(df.sub[df.sub$diagnosis == "BOTH",])))
COMP     = sprintf("%.2f (±%.2f)",
                   mean(df.sub[df.sub$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, `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 == "BOTH",]$ASRS_total),
            sd(df.sub[df.sub$diagnosis == "BOTH",]$ASRS_total)/
              sqrt(nrow(df.sub[df.sub$diagnosis == "BOTH",]))),
    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 == "BOTH",]$BDI_total),
            sd(df.sub[df.sub$diagnosis == "BOTH",]$BDI_total)/
              sqrt(nrow(df.sub[df.sub$diagnosis == "BOTH",]))),
    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$diagnosis == "COMP",]))),
    sprintf("%.3f", aov.BDI@bayesFactor[["bf"]])
  ),
  c(
    "Gender (diverse/agender - female - male)",
    sprintf("%d - %d - %d",
            nrow(df.sub[df.sub$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 == "BOTH" & df.sub$gender == "dan",]),
            nrow(df.sub[df.sub$diagnosis == "BOTH" & df.sub$gender == "fem",]),
            nrow(df.sub[df.sub$diagnosis == "BOTH" & 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$diagnosis == "BOTH",]$iq),
              sd(df.sub[df.sub$diagnosis == "BOTH",]$iq)/
                sqrt(nrow(df.sub[df.sub$diagnosis == "BOTH",]))),
      sprintf("%.2f (±%.2f)",
              mean(df.sub[df.sub$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(
      "RADS-R",
      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$diagnosis == "BOTH",]$RAADS_total),
              sd(df.sub[df.sub$diagnosis == "BOTH",]$RAADS_total)/
                sqrt(nrow(df.sub[df.sub$diagnosis == "BOTH",]))),
      sprintf("%.2f (±%.2f)",
              mean(df.sub[df.sub$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$diagnosis == "BOTH",]$TAS_total),
              sd(df.sub[df.sub$diagnosis == "BOTH",]$TAS_total)/
```

```
                    sqrt(nrow(df.sub[df.sub$diagnosis == "BOTH",]))),
          sprintf("%.2f (±%.2f)",
                  mean(df.sub[df.sub$diagnosis == "COMP",]$TAS_total),
                  sd(df.sub[df.sub$diagnosis == "COMP",]$TAS_total)/
                    sqrt(nrow(df.sub[df.sub$diagnosis == "COMP",]))),
          sprintf("%.3f", aov.TAS@bayesFactor[["bf"]])
        )
  ) %>% arrange(measurement)

  # 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
    )

  # calculate variance of reaction times:
  # no difficulty due to suboptimal posterior fit
  df.var = df.tsk %>%
    group_by(subID, diagnosis, phase, expected) %>%
    summarise(
      totaln = n(),
      valuen = sum(!is.na(rt.cor)),
      rt.var = sd(rt.cor, na.rm = T)
    ) %>%
    mutate(
      perc = valuen/totaln
    ) %>% filter(perc >= 2/3) %>%
    mutate_if(is.character, as.factor)

  # 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, gen.desc, tb.gen,
       file = "data/PAL_data.RData")

} else {

  load("data/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	
BOTH	ASRS_total	0.9547574	0.3660251	
COMP	ASRS_total	0.9068414	0.0408291	*
ADHD	BDI_total	0.8019340	0.0005365	*
ASD	BDI_total	0.8511614	0.0028558	*
BOTH	BDI_total	0.8324214	0.0013277	*
COMP	BDI_total	0.7469562	0.0000825	*
ADHD	RAADS_total	0.9270567	0.1065426	
ASD	RAADS_total	0.9556216	0.3808761	
BOTH	RAADS_total	0.9524787	0.3291283	
COMP	RAADS_total	0.8644129	0.0061719	*
ADHD	TAS_total	0.9626930	0.5455600	
ASD	TAS_total	0.9240264	0.0812916	
BOTH	TAS_total	0.9445280	0.2245560	
COMP	TAS_total	0.8719909	0.0085270	*
ADHD	age	0.9439542	0.2386038	
ASD	age	0.9396864	0.1769487	
BOTH	age	0.9503886	0.2981103	
COMP	age	0.8212328	0.0010965	*
ADHD	iq	0.9736733	0.7942630	
ASD	iq	0.9591210	0.4458484	
BOTH	iq	0.9583546	0.4309600	
COMP	iq	0.9589488	0.4683259	

```
rm(df.sht)

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

```
##                          bf error                      time          code
## Non-indep. (a=1) -4.442386     0 Mon Feb  3 13:46:09 2025 228b5572a9d89
```

```
# only male and female participants
ct.mf@bayesFactor
```

```
##                           bf error                       time          code
## Non-indep. (a=1) -2.758595     0 Mon Feb  3 13:46:09 2025 228b56df2acd6
```

```
# print descriptions in not-female and not-male group > remove umlaute
print(gsub("[[:punct:]]", "", gen.desc))
```

```
## [1] "nicht binr" "divers"     "agender"
```

```
# print gender and trans/cis distribution
kable(tb.gen)
```

gender	diagnosis	cis	Freq
dan	ADHD	cis	0
fem	ADHD	cis	9
mal	ADHD	cis	10
dan	ASD	cis	0
fem	ASD	cis	11
mal	ASD	cis	11
dan	BOTH	cis	0
fem	BOTH	cis	12
mal	BOTH	cis	6
dan	COMP	cis	0
fem	COMP	cis	10
mal	COMP	cis	12
dan	ADHD	trans	2
fem	ADHD	trans	0
mal	ADHD	trans	1
dan	ASD	trans	0
fem	ASD	trans	0
mal	ASD	trans	1
dan	BOTH	trans	3
fem	BOTH	trans	0
mal	BOTH	trans	2
dan	COMP	trans	0
fem	COMP	trans	0
mal	COMP	trans	0

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

```
##      [,1] [,2] [,3]
## ADHD    1    0    0
## ASD     0    1    0
```

```
## BOTH    0    0    1
## COMP   -1   -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$diagnosis) = contr.sum(4)
contrasts(df.var$diagnosis)
```

```
##      [,1] [,2] [,3]
## ADHD    1    0    0
## ASD     0    1    0
## BOTH    0    0    1
## COMP   -1   -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	ADHD.ASD	COMP	logBF10
ASRS	43.95 (±2.59)	32.39 (±1.71)	46.43 (±1.94)	25.95 (±1.70)	19.617
Age	26.23 (±1.41)	29.35 (±1.64)	30.22 (±1.72)	27.59 (±1.25)	-1.489

measurement	ADHD	ASD	ADHD.ASD	COMP	logBF10
BDI	7.95 (±1.78)	10.22 (±1.74)	8.61 (±1.71)	2.32 (±0.68)	6.513
Gender (diverse/agender - female - male)	2 - 9 - 11	0 - 11 - 12	3 - 12 - 8	0 - 10 - 12	-4.442
IQ	108.57 (±2.39)	113.33 (±3.18)	112.93 (±2.34)	109.14 (±1.84)	-1.745
RADS-R	93.95 (±8.56)	150.26 (±8.37)	146.52 (±6.97)	46.50 (±7.68)	27.807
TAS	50.82 (±2.49)	62.61 (±1.31)	56.48 (±2.15)	39.36 (±2.07)	18.389

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
2	expected	3
2	unexpected	3
3	expected	3
3	unexpected	3

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

subID	phase	n
1	prevolatile	2
1	volatile	2
1	postvolatile	2
2	prevolatile	2
2	volatile	2
2	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")))
}
```

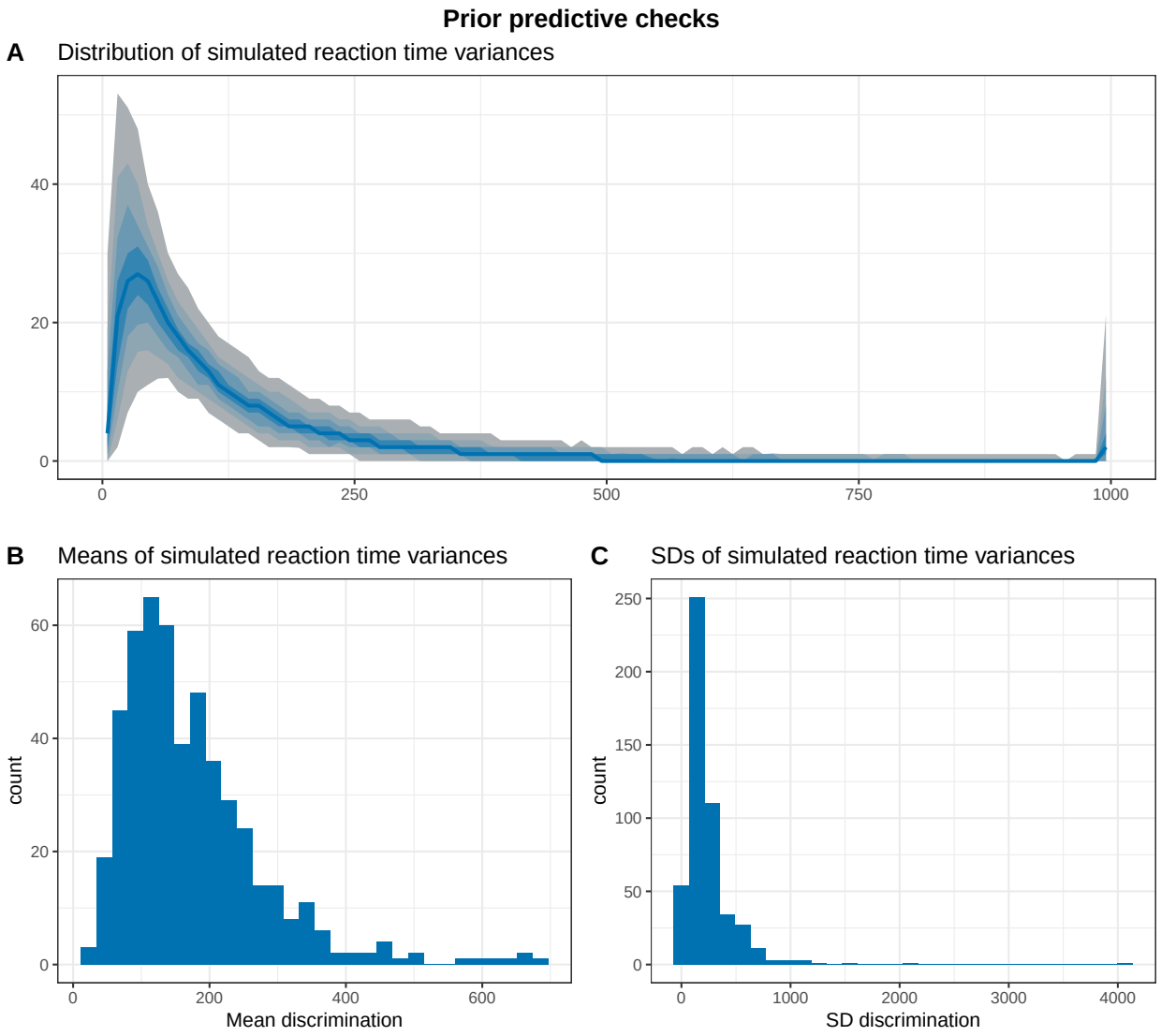
We start by investigating the rhats and the number of divergent samples. This shows that only 0 of 500 simulations had at least one parameter that had an rhat of at least 1.05 and 5 models had divergent samples.

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)$counts
}
# for each bin, compute quantiles across histograms
probs = seq(0.1, 0.9, 0.1)
quantmat = as.data.frame(matrix(NA, nrow=dim(histmat)[1], ncol = length(probs)))
names(quantmat) = paste0("p", probs)
for (i in 1:dim(histmat)[1]) {
  quantmat[i,] = quantile(histmat[i,], p = probs)
}
quantmat$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")
```

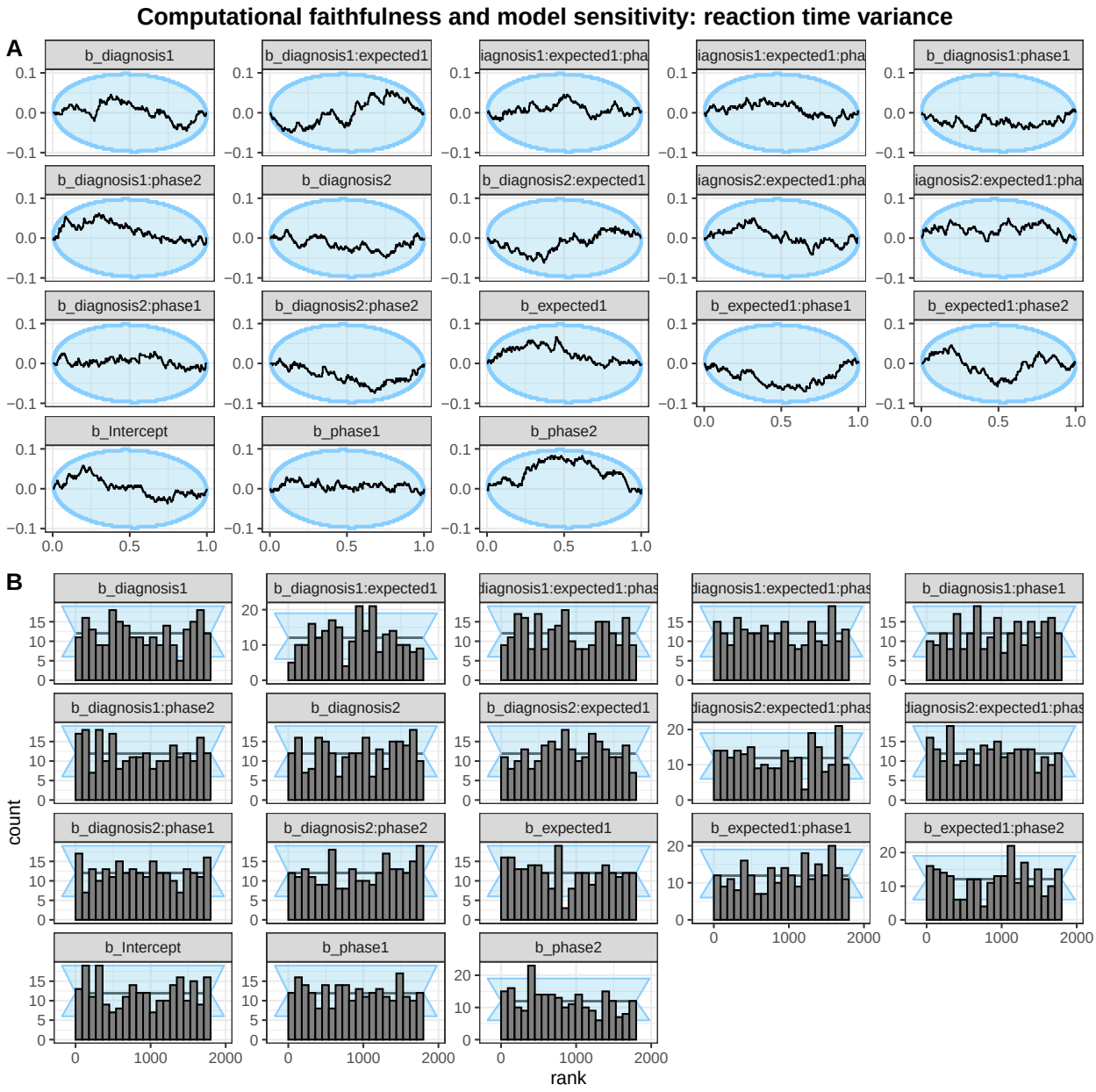
```
annotate_figure(p, top = text_grob("Prior predictive checks", face = "bold", size = 14))
```



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)
```

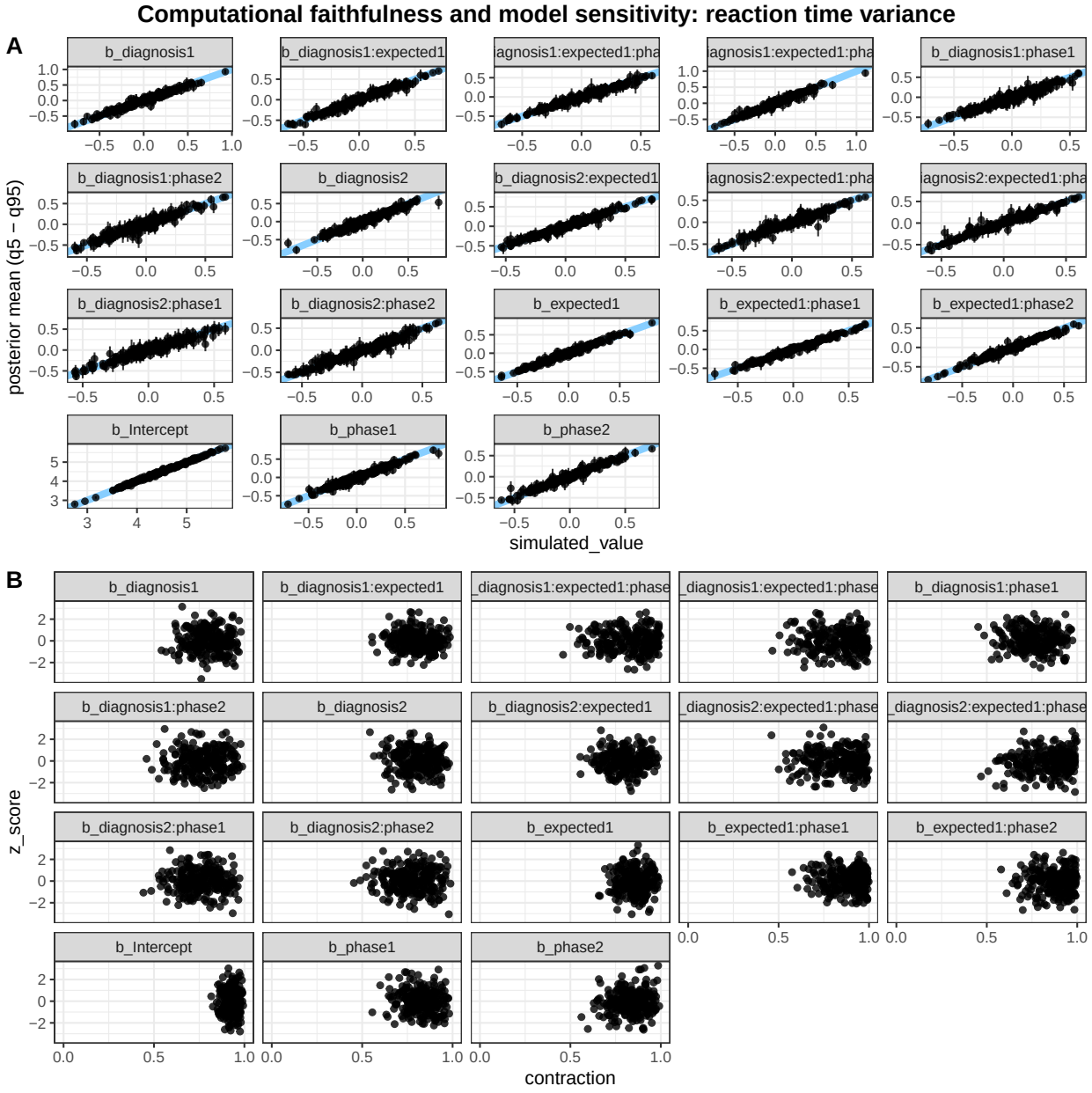
```r
# plot SBC with functions from the SBC package focusing on population-level parameters
a = 0.5
df.results.b = df.results %>%
  filter(substr(variable, 1, 2) == "b_") %>%
  filter(!(sim_id %in% check$sim_id))
p1 = plot_ecdf_diff(df.results.b) + theme_bw() + theme(legend.position = "none") +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 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 vari
```



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 = .8) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p4 = plot_contraction(df.results.b, prior_sd = setNames(c(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 vari
```



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.

```
# 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 = file.path(brms_dir, "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:
```

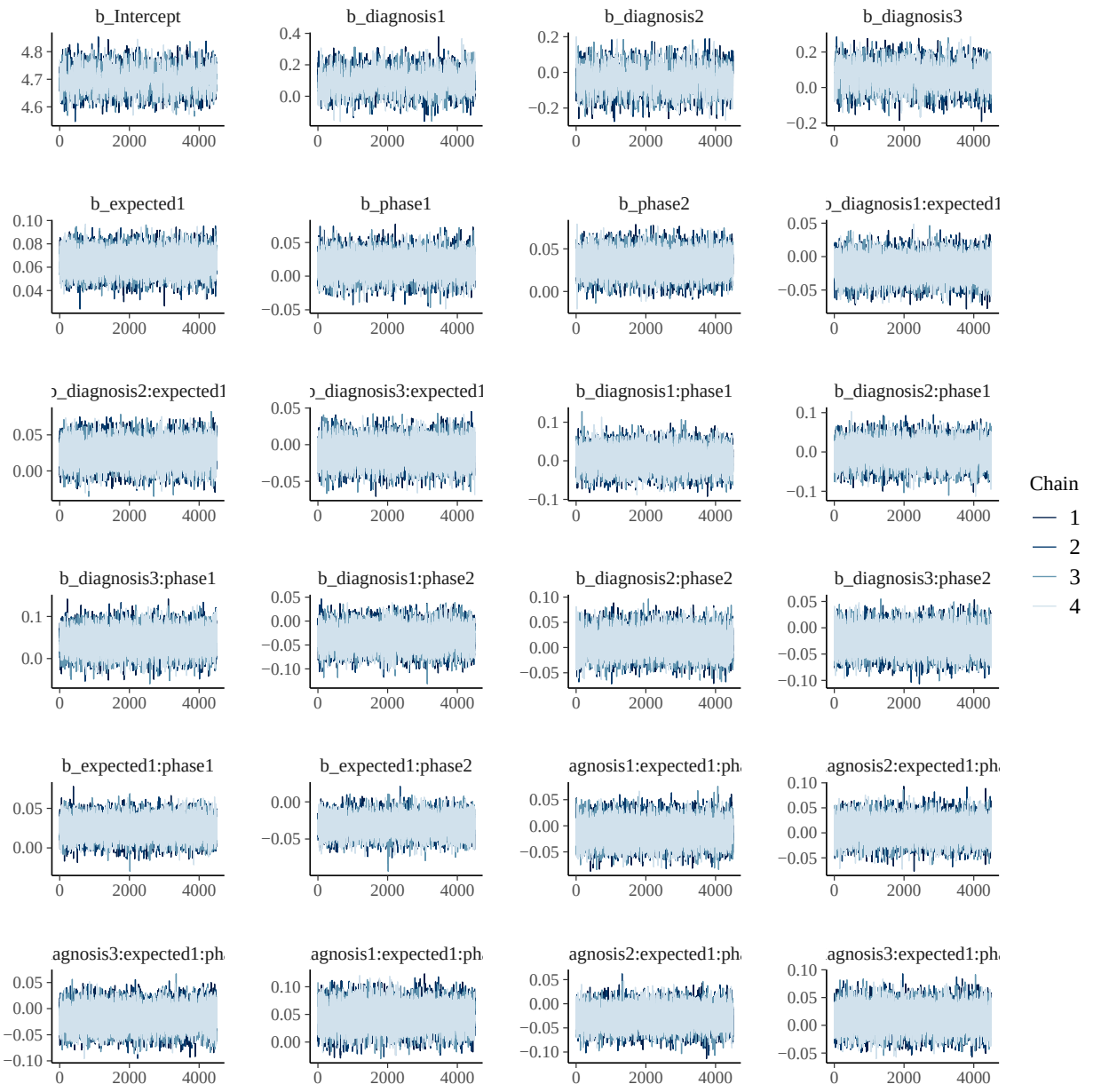
```
## E-BFMI indicated no pathological behavior.
```

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

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

```
# 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))
```

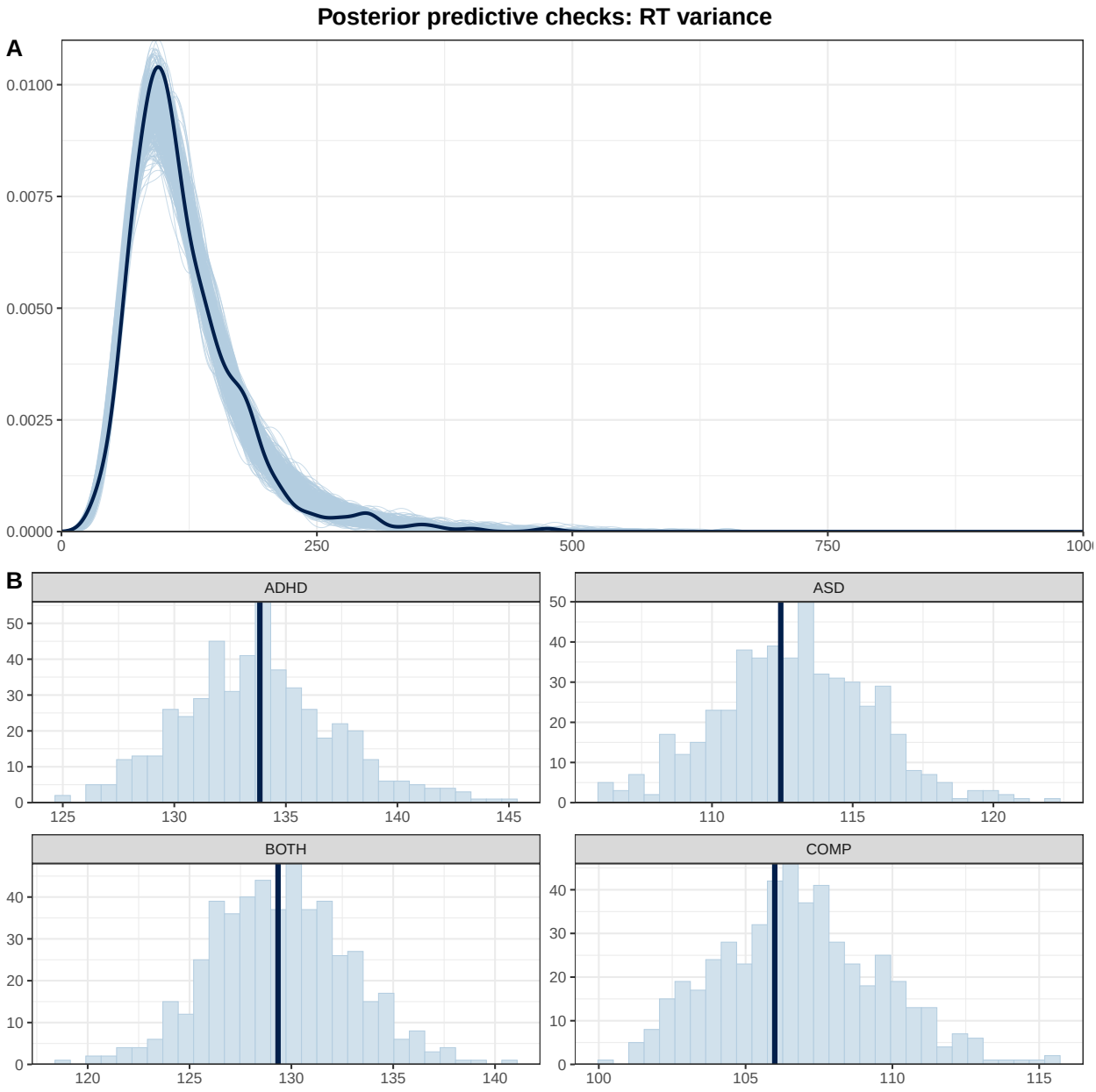
```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```



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: RT variance", face = "bold", size = 14)
```



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.

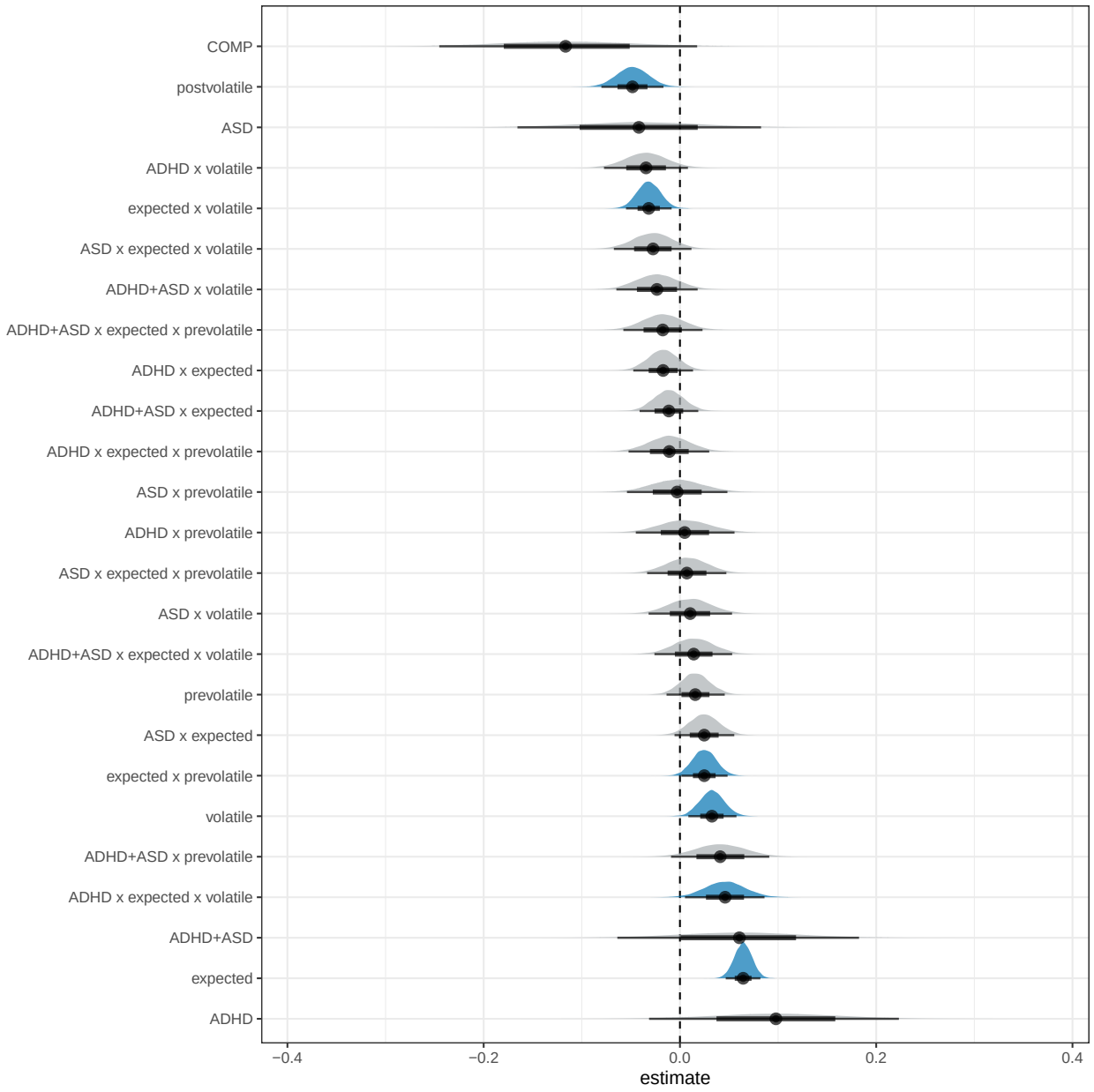
```r
# print a summary
summary(m.var)
```

```
##  Family: lognormal
##   Links: mu = identity; sigma = identity
## Formula: rt.var ~ diagnosis * expected * phase + (expected + phase | subID)
##    Data: df.var (Number of observations: 523)
##   Draws: 4 chains, each with iter = 6000; warmup = 1500; thin = 1;
##          total post-warmup draws = 18000
##
## Multilevel Hyperparameters:
## ~subID (Number of levels: 90)
##                           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)                 0.35      0.03     0.31     0.41 1.00     3239
## sd(expected1)                 0.03      0.01     0.00     0.05 1.00     3452
## sd(phase1)                    0.08      0.02     0.05     0.12 1.00     5911
## sd(phase2)                    0.03      0.02     0.00     0.07 1.00     3663
## cor(Intercept,expected1)     -0.41      0.30    -0.87     0.30 1.00    14682
## cor(Intercept,phase1)        -0.21      0.17    -0.54     0.14 1.00    12773
## cor(expected1,phase1)         0.38      0.34    -0.40     0.90 1.00     1445
## cor(Intercept,phase2)        -0.04      0.33    -0.69     0.64 1.00    17825
## cor(expected1,phase2)         0.33      0.41    -0.60     0.91 1.00     4350
## cor(phase1,phase2)            0.28      0.37    -0.53     0.88 1.00    12250
##                           Tail_ESS
## sd(Intercept)                 6450
## sd(expected1)                 3281
## sd(phase1)                    5713
## sd(phase2)                    5471
## cor(Intercept,expected1)      8528
## cor(Intercept,phase1)        10789
## cor(expected1,phase1)         2579
## cor(Intercept,phase2)        10007
## cor(expected1,phase2)         7736
## cor(phase1,phase2)           12154
##
## Regression Coefficients:
##                             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                       4.70      0.04     4.62     4.77 1.00     1765
## diagnosis1                      0.10      0.06    -0.03     0.22 1.00     2067
## diagnosis2                     -0.04      0.06    -0.17     0.08 1.00     2090
## diagnosis3                      0.06      0.06    -0.06     0.18 1.00     2256
## expected1                       0.06      0.01     0.05     0.08 1.00    22169
## phase1                          0.02      0.02    -0.01     0.05 1.00    15062
## phase2                          0.03      0.01     0.01     0.06 1.00    22768
## diagnosis1:expected1           -0.02      0.02    -0.05     0.01 1.00    20006
## diagnosis2:expected1            0.02      0.02    -0.01     0.06 1.00    18764
## diagnosis3:expected1           -0.01      0.02    -0.04     0.02 1.00    18730
## diagnosis1:phase1               0.01      0.03    -0.04     0.06 1.00    12597
## diagnosis2:phase1              -0.00      0.03    -0.05     0.05 1.00    13486
## diagnosis3:phase1               0.04      0.03    -0.01     0.09 1.00    11683
## diagnosis1:phase2              -0.03      0.02    -0.08     0.01 1.00    17888
## diagnosis2:phase2               0.01      0.02    -0.03     0.05 1.00    18854
## diagnosis3:phase2              -0.02      0.02    -0.06     0.02 1.00    18348
## expected1:phase1                0.02      0.01     0.00     0.05 1.00    22900
```

```
## expected1:phase2                -0.03      0.01    -0.05    -0.01 1.00    22647
## diagnosis1:expected1:phase1     -0.01      0.02    -0.05     0.03 1.00    16499
## diagnosis2:expected1:phase1      0.01      0.02    -0.03     0.05 1.00    16989
## diagnosis3:expected1:phase1     -0.02      0.02    -0.06     0.02 1.00    17984
## diagnosis1:expected1:phase2      0.05      0.02     0.01     0.09 1.00    16549
## diagnosis2:expected1:phase2     -0.03      0.02    -0.07     0.01 1.00    18130
## diagnosis3:expected1:phase2      0.01      0.02    -0.03     0.05 1.00    17244
##                              Tail_ESS
## Intercept                        3952
## diagnosis1                       3789
## diagnosis2                       3737
## diagnosis3                       4254
## expected1                       13632
## phase1                          14428
## phase2                          12519
## diagnosis1:expected1            13959
## diagnosis2:expected1            14598
## diagnosis3:expected1            14547
## diagnosis1:phase1               13028
## diagnosis2:phase1               13990
## diagnosis3:phase1               13236
## diagnosis1:phase2               14261
## diagnosis2:phase2               14368
## diagnosis3:phase2               15061
## expected1:phase1                14419
## expected1:phase2                14607
## diagnosis1:expected1:phase1     14031
## diagnosis2:expected1:phase1     14225
## diagnosis3:expected1:phase1     14514
## diagnosis1:expected1:phase2     13412
## diagnosis2:expected1:phase2     14890
## diagnosis3:expected1:phase2     15300
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.19      0.01     0.18     0.21 1.00     6449    10684
##
## 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_diagnosis3,
    b_postvolatile = - b_phase1 - b_phase2,
    ASD            = b_Intercept + b_diagnosis2,
    BOTH           = b_Intercept + b_diagnosis3,
    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, "diagnosis3", "ADHD+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 + diagnosis3")
h1a
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis1... < 0    -0.21      0.11    -0.39    -0.03      39.36
##   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.
```

```r
## 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
    )
```

[!MISSING]

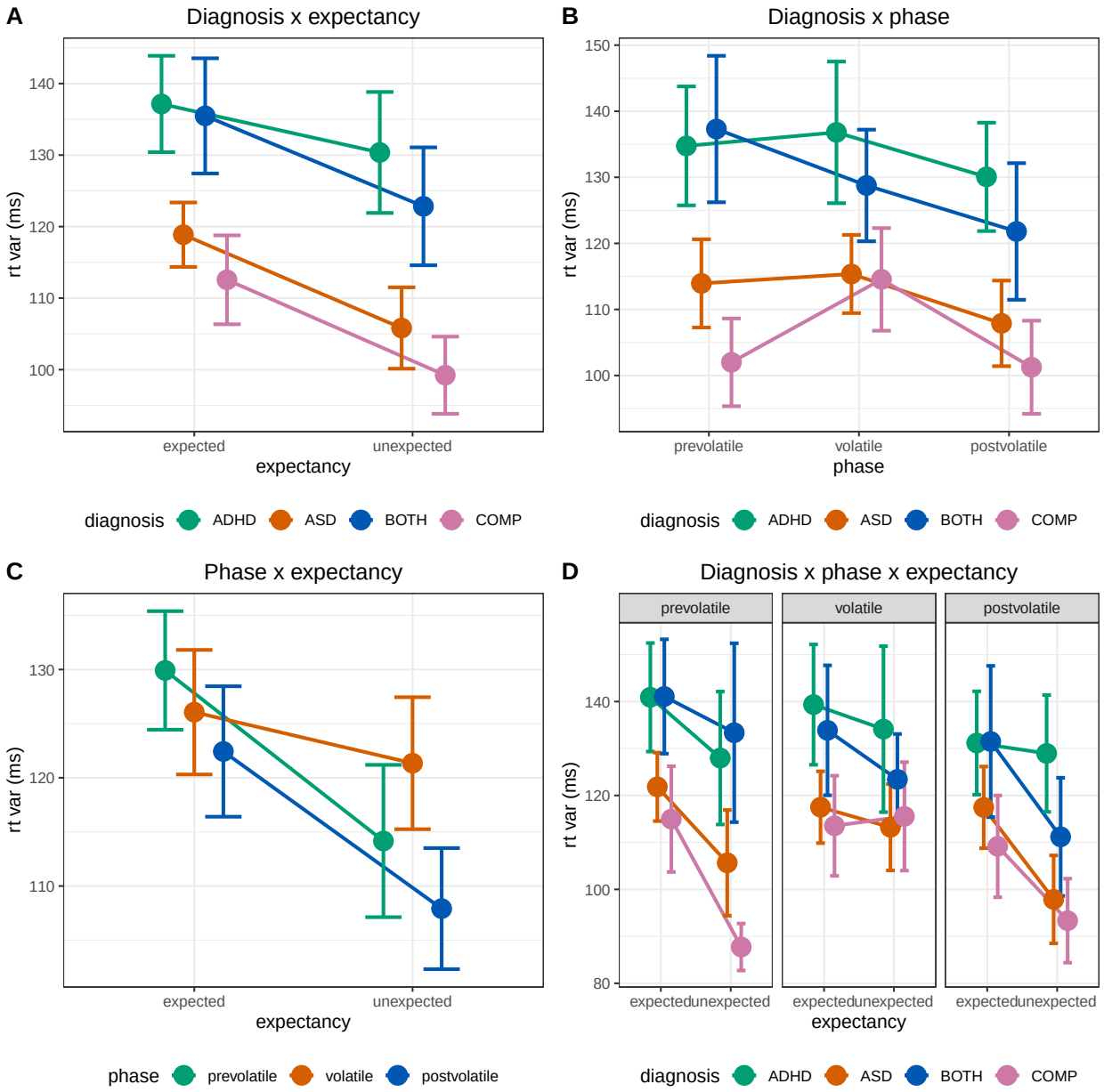
Plots

```r
# rain cloud plot

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 = .8),
violin.args = list(color = "black", outlier.shape = NA, alpha = .8),
boxplot.args.pos = list(
  position = ggpp::position_dodgenudge(x = 0, width = 0.3), width = 0.3
),
point.args = list(show_guide = FALSE, alpha = .5),
violin.args.pos = list(
  width = 0.6, position = position_nudge(x = 0.16)),
point.args.pos = list(position = ggpp::position_dodgenudge(x = -0.25, width = 0.1))) +
  scale_fill_manual(values = custom.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.

```
# 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")
```



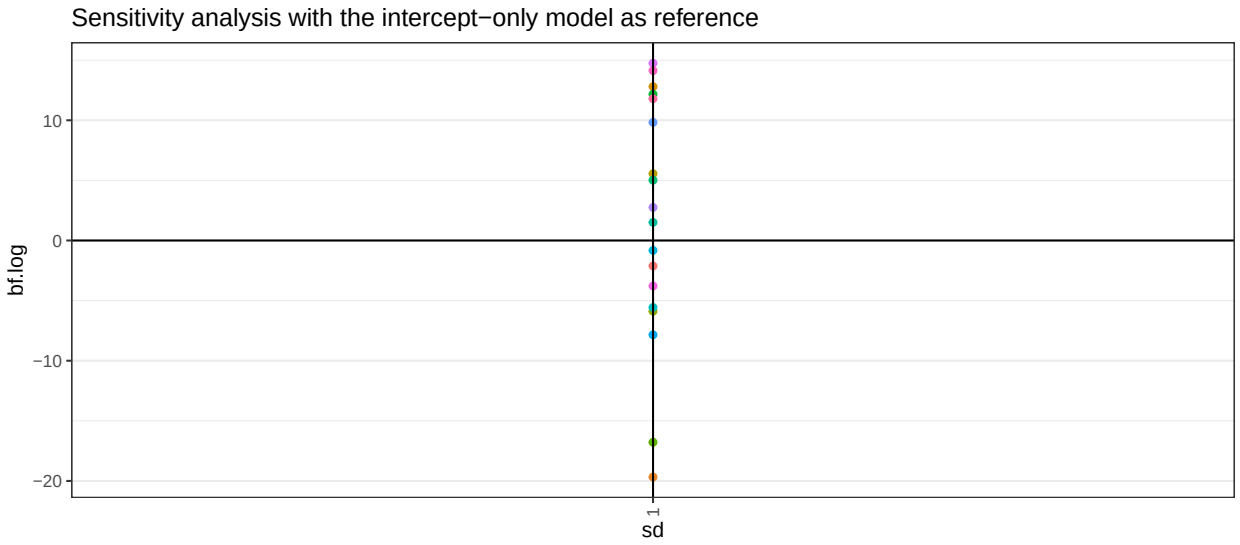
Bayes factor analysis

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 = 40000, warmup = 10000,
            backend = "cmdstanr", threads = threading(8),
            file = file.path(brms_dir, "m_var_bf"),
            save_pars = save_pars(all = TRUE)
            )

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

# 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	14.751
phase + expected	14.132
diagnosis + expected	12.793
diagnosis + phase + expected	12.166
phase + expected + phase:expected	11.795
diagnosis + phase + expected + phase:expected	9.826

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. We tried both this full model and one without difficulty:

rt.cor ~ diagnosis * expected * phase * difficulty + (expected * phase * difficulty | subID) + (diagnosis * expected * phase * difficulty | trl), list(), list(), rtcor, NULL

rt.cor ~ diagnosis * expected * phase + (expected * phase | subID) + (diagnosis * expected * phase | trl), list(), list(), rtcor, NULL

Unfortunately, they both showed suboptimal posterior fit. They are uploaded to GitHub and can be inspected.

[!CHECK] 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), # higher due to aggregation
  # 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)
)
```

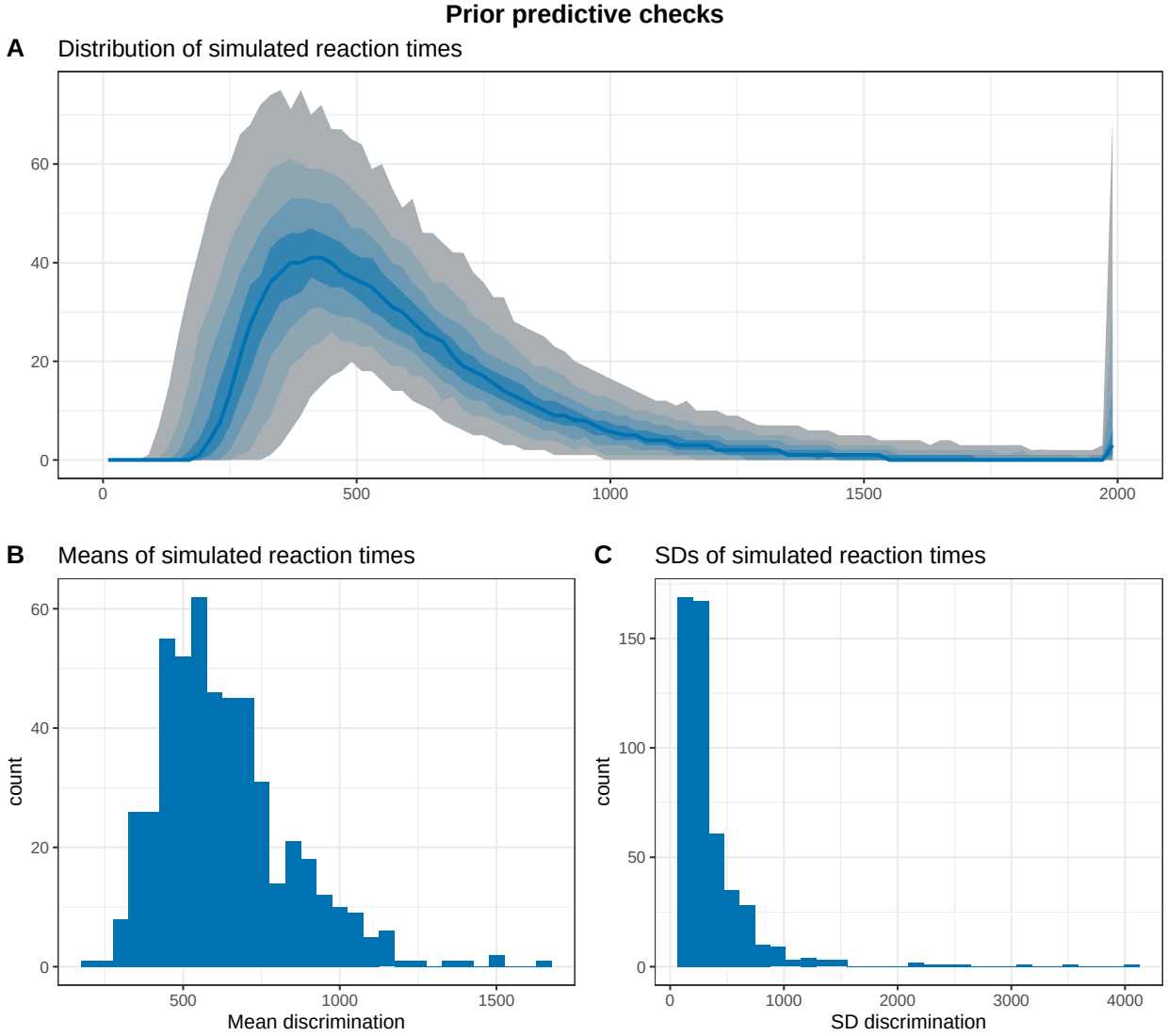
Simulation-based calibration

```
if (!(file.exists(file.path(cache_dir, sprintf("df_res_%s.rds", code)))) |
    !(file.exists(file.path(cache_dir, sprintf("df_res_%s.rds", code))))) {
  # 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,
```

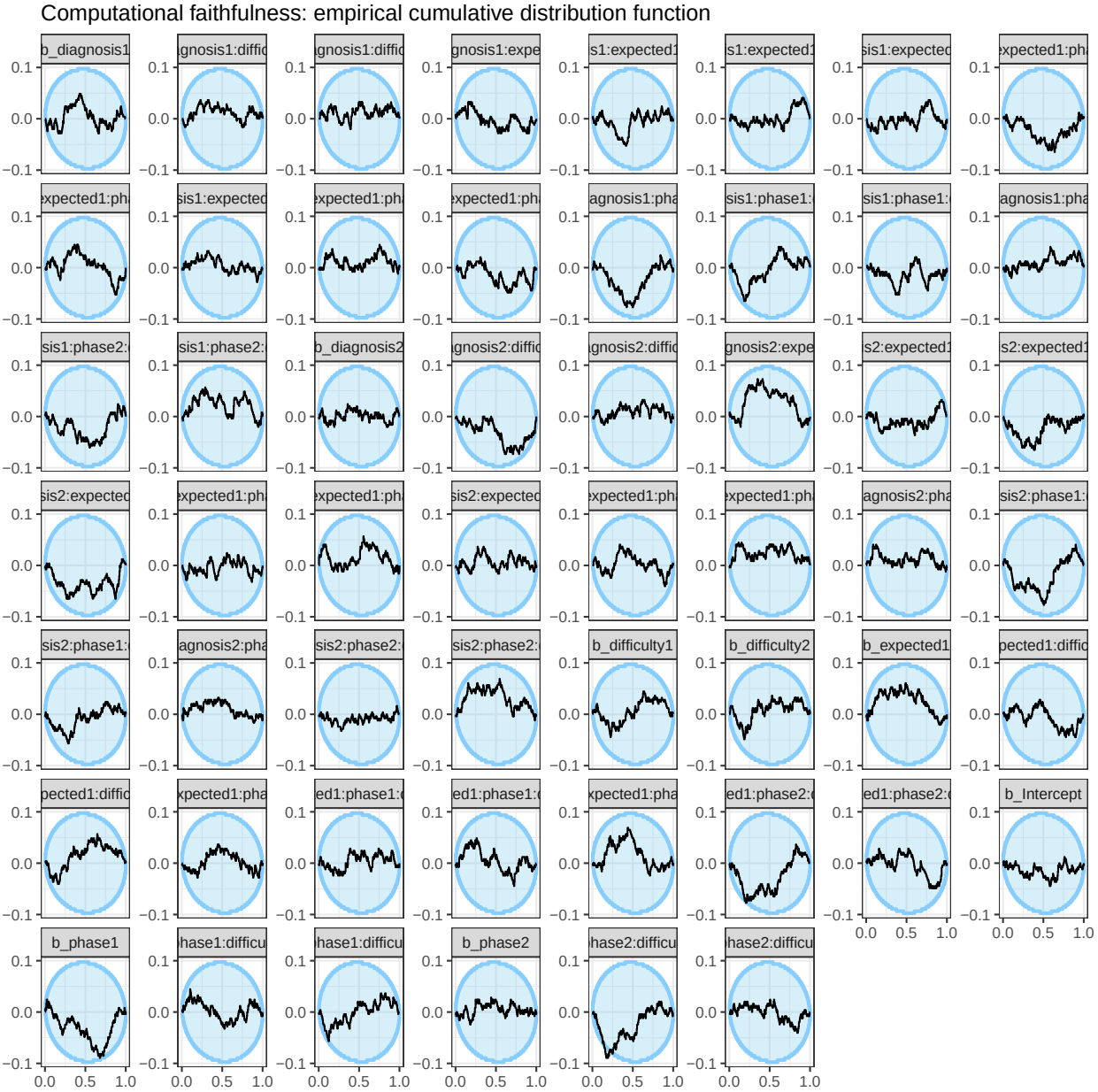
```
  ggarrange(p2, p3, ncol = 2, labels = c("B", "C")),
  nrow = 2, labels = "A")
annotate_figure(p, top = text_grob("Prior predictive checks", face = "bold", size = 14))
```



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.

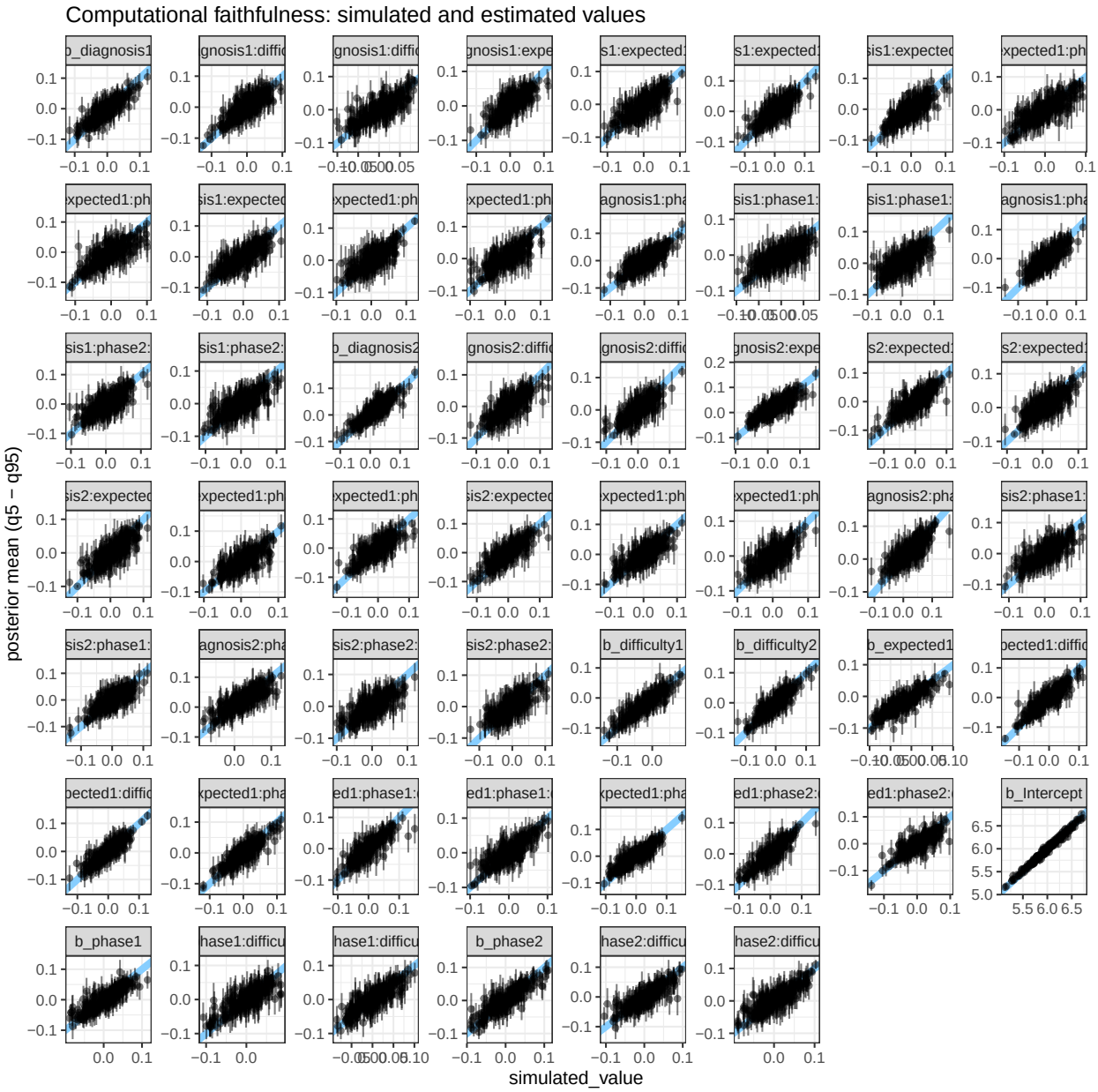
```
# get simulation numbers with issues
rank = max(df.results$max_rank)
check = merge(df.results %>%
    group_by(sim_id) %>%
      summarise(
        rhat = max(rhat, na.rm = T),
        mean_rank = mean(max_rank)
        ) %>%
```

```
    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")
```

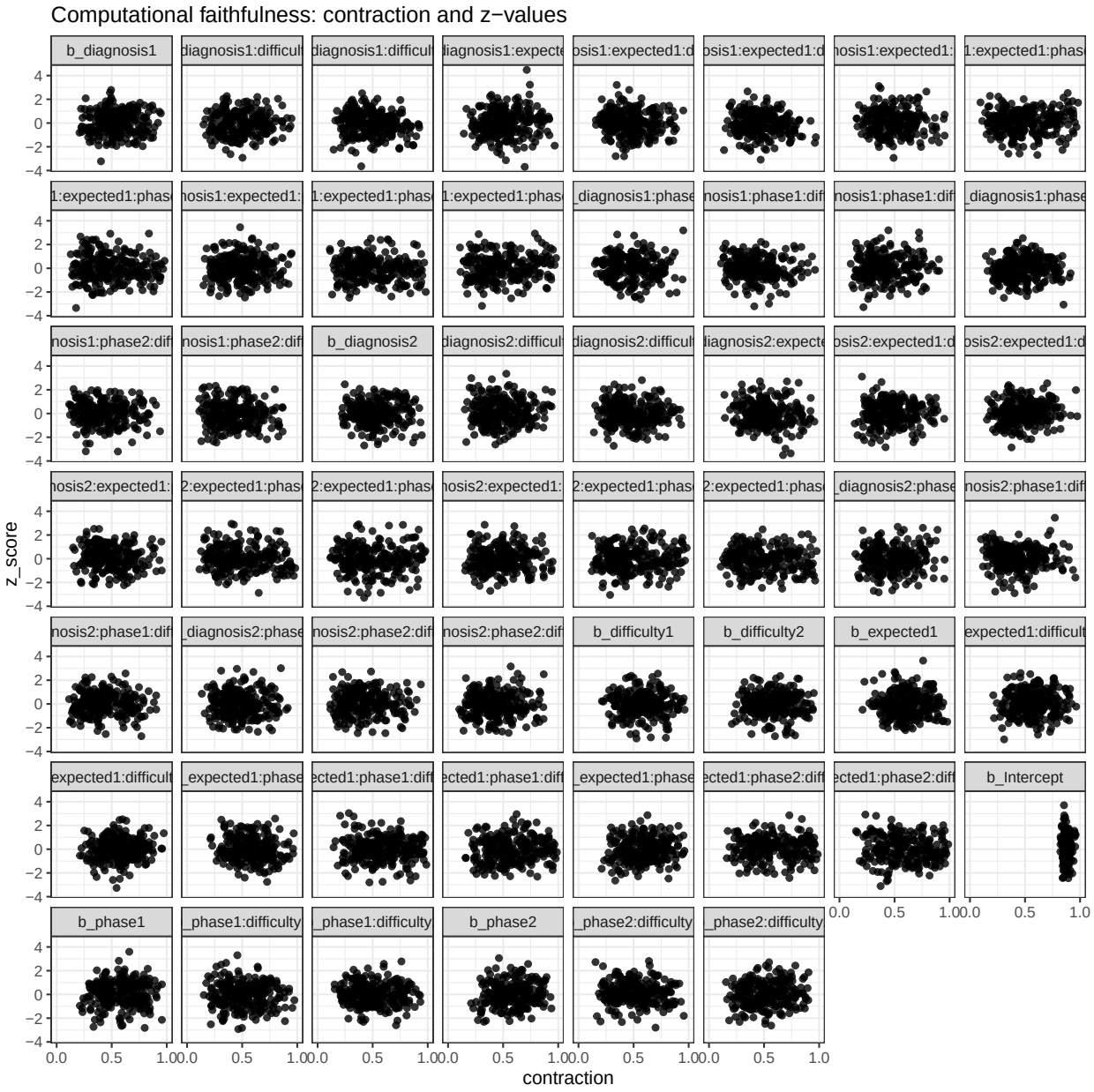


```
plot_rank_hist(df.results.b, bins = 20) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3)) +
  ggtitle("Computational faithfulness: rank histograms")
```


```
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.

After having run the SBC, we realised that setting the threshold to Gaussian with a mean of 100ms does not fit well with a model based on aggregated reaction times - they are unlikely to be this low. Therefore, we

change this one prior to a 200ms mean, but assume that the SBC still holds.

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.

```
# adjust ndt prior
priors = priors %>%
  mutate(
    prior = if_else(class == "ndt", "normal(250, 100)", prior)
  )

# 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 = file.path(brms_dir, "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
```

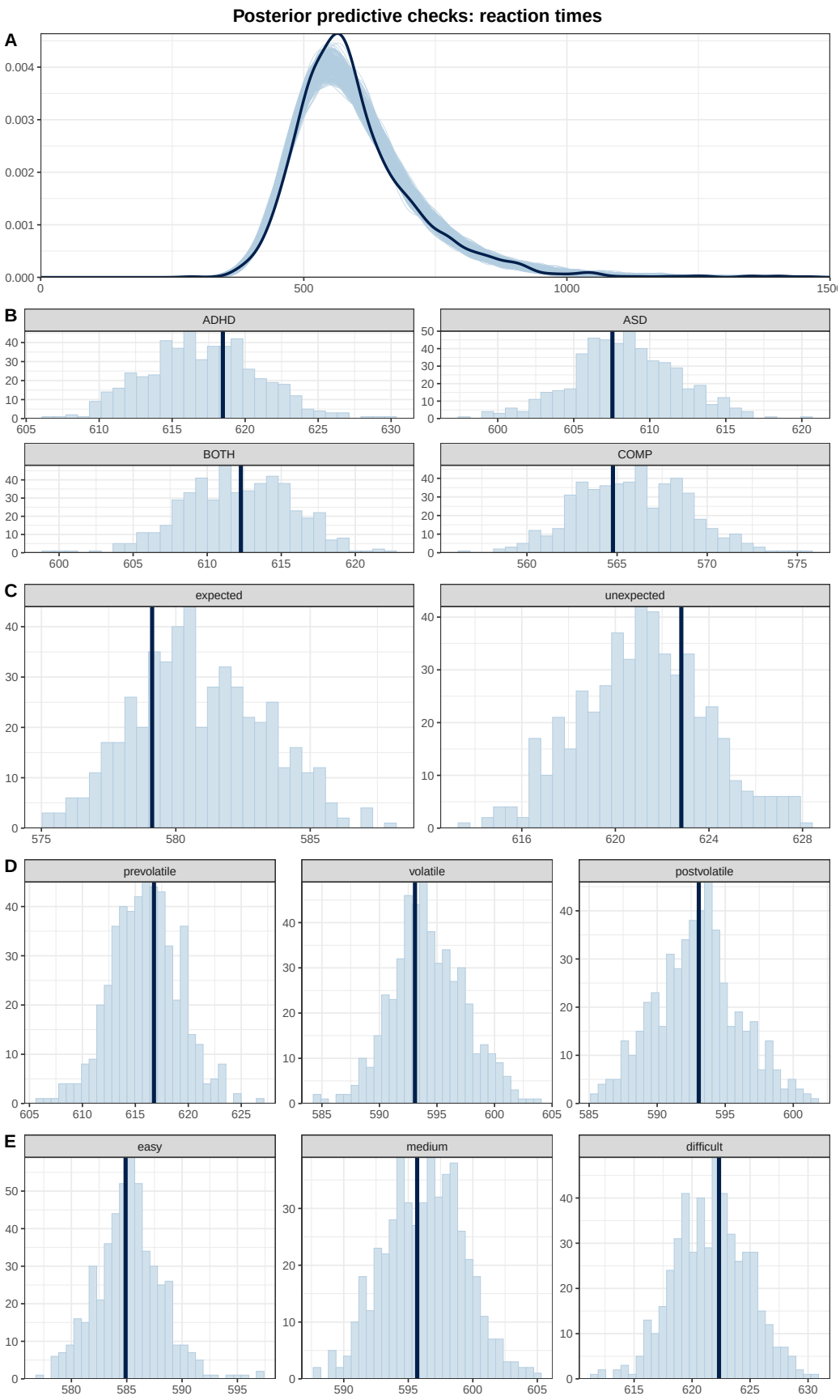
```
# check the trace plots
post.draws = as_draws_df(m.pal)
mcmc_trace(post.draws, regex_pars = "^b_",
           facet_args = list(ncol = 6)) +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
```

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


b_Intercept
b_phase1
b_phase2
Chain
1
2
3
4

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
BOTH
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: 1620)
##   Draws: 4 chains, each with iter = 6000; warmup = 1500; thin = 1;
##          total post-warmup draws = 18000
##
## Multilevel Hyperparameters:
## ~subID (Number of levels: 90)
##                                                 Estimate Est.Error l-95% CI
## sd(Intercept)                                       0.23      0.02     0.20
## sd(expected1)                                       0.03      0.00     0.02
## sd(phase1)                                          0.08      0.01     0.07
## sd(phase2)                                          0.03      0.01     0.02
## sd(difficulty1)                                     0.01      0.01     0.00
## sd(difficulty2)                                     0.02      0.01     0.01
## sd(expected1:phase1)                                0.04      0.01     0.02
## sd(expected1:phase2)                                0.03      0.01     0.01
## 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     0.01
## cor(Intercept,expected1)                            0.02      0.13    -0.23
## cor(Intercept,phase1)                               0.12      0.10    -0.09
## cor(expected1,phase1)                               0.20      0.14    -0.09
## cor(Intercept,phase2)                              -0.16      0.14    -0.43
## cor(expected1,phase2)                               0.19      0.17    -0.16
## cor(phase1,phase2)                                 -0.32      0.14    -0.56
## cor(Intercept,difficulty1)                         -0.11      0.21    -0.50
## cor(expected1,difficulty1)                         -0.06      0.22    -0.49
## cor(phase1,difficulty1)                            -0.03      0.21    -0.44
## cor(phase2,difficulty1)                            -0.08      0.22    -0.50
## cor(Intercept,difficulty2)                         -0.18      0.17    -0.50
## cor(expected1,difficulty2)                         -0.24      0.20    -0.60
## cor(phase1,difficulty2)                            -0.28      0.18    -0.60
## cor(phase2,difficulty2)                            -0.08      0.20    -0.45
## cor(difficulty1,difficulty2)                       -0.06      0.24    -0.51
## cor(Intercept,expected1:phase1)                    -0.10      0.14    -0.36
## cor(expected1,expected1:phase1)                     0.18      0.17    -0.16
## cor(phase1,expected1:phase1)                       -0.34      0.14    -0.59
## cor(phase2,expected1:phase1)                        0.06      0.18    -0.29
```

```
## cor(difficulty1,expected1:phase1)                  -0.06      0.22    -0.48
## cor(difficulty2,expected1:phase1)                   0.03      0.20    -0.36
## cor(Intercept,expected1:phase2)                     0.38      0.14     0.08
## cor(expected1,expected1:phase2)                    -0.16      0.18    -0.50
## cor(phase1,expected1:phase2)                        0.06      0.17    -0.27
## cor(phase2,expected1:phase2)                       -0.49      0.16    -0.76
## cor(difficulty1,expected1:phase2)                  -0.01      0.22    -0.44
## cor(difficulty2,expected1:phase2)                   0.11      0.20    -0.29
## cor(expected1:phase1,expected1:phase2)             -0.14      0.19    -0.48
## cor(Intercept,phase1:difficulty1)                  -0.26      0.14    -0.53
## cor(expected1,phase1:difficulty1)                  -0.15      0.18    -0.49
## cor(phase1,phase1:difficulty1)                      0.09      0.15    -0.21
## cor(phase2,phase1:difficulty1)                     -0.03      0.19    -0.39
## cor(difficulty1,phase1:difficulty1)                 0.10      0.23    -0.37
## cor(difficulty2,phase1:difficulty1)                 0.00      0.21    -0.40
## cor(expected1:phase1,phase1:difficulty1)           -0.14      0.18    -0.48
## cor(expected1:phase2,phase1:difficulty1)           -0.12      0.19    -0.49
## cor(Intercept,phase2:difficulty1)                  -0.07      0.19    -0.42
## cor(expected1,phase2:difficulty1)                  -0.21      0.21    -0.59
## cor(phase1,phase2:difficulty1)                     -0.04      0.20    -0.42
## cor(phase2,phase2:difficulty1)                     -0.21      0.21    -0.59
## cor(difficulty1,phase2:difficulty1)                -0.00      0.23    -0.44
## cor(difficulty2,phase2:difficulty1)                 0.17      0.22    -0.29
## cor(expected1:phase1,phase2:difficulty1)            0.04      0.21    -0.38
## cor(expected1:phase2,phase2:difficulty1)            0.22      0.21    -0.24
## cor(phase1:difficulty1,phase2:difficulty1)          0.02      0.22    -0.39
## cor(Intercept,phase1:difficulty2)                   0.12      0.18    -0.25
## cor(expected1,phase1:difficulty2)                   0.20      0.21    -0.22
## cor(phase1,phase1:difficulty2)                      0.00      0.19    -0.37
## cor(phase2,phase1:difficulty2)                      0.21      0.21    -0.22
## cor(difficulty1,phase1:difficulty2)                -0.12      0.24    -0.56
## cor(difficulty2,phase1:difficulty2)                -0.16      0.22    -0.57
## cor(expected1:phase1,phase1:difficulty2)           -0.04      0.21    -0.43
## cor(expected1:phase2,phase1:difficulty2)           -0.15      0.21    -0.54
## cor(phase1:difficulty1,phase1:difficulty2)         -0.17      0.22    -0.56
## cor(phase2:difficulty1,phase1:difficulty2)         -0.20      0.23    -0.60
## cor(Intercept,phase2:difficulty2)                   0.23      0.16    -0.10
## cor(expected1,phase2:difficulty2)                   0.17      0.19    -0.23
## cor(phase1,phase2:difficulty2)                     -0.06      0.17    -0.39
## cor(phase2,phase2:difficulty2)                      0.10      0.20    -0.30
## cor(difficulty1,phase2:difficulty2)                -0.04      0.23    -0.49
## cor(difficulty2,phase2:difficulty2)                -0.18      0.21    -0.57
## cor(expected1:phase1,phase2:difficulty2)            0.09      0.19    -0.30
## cor(expected1:phase2,phase2:difficulty2)            0.04      0.20    -0.36
## cor(phase1:difficulty1,phase2:difficulty2)         -0.23      0.20    -0.59
## cor(phase2:difficulty1,phase2:difficulty2)         -0.19      0.23    -0.60
## cor(phase1:difficulty2,phase2:difficulty2)          0.13      0.22    -0.31
## cor(Intercept,expected1:difficulty1)                0.05      0.22    -0.39
## cor(expected1,expected1:difficulty1)                0.05      0.23    -0.41
## cor(phase1,expected1:difficulty1)                   0.04      0.23    -0.41
## cor(phase2,expected1:difficulty1)                   0.04      0.23    -0.42
## cor(difficulty1,expected1:difficulty1)             -0.12      0.25    -0.59
## cor(difficulty2,expected1:difficulty1)             -0.03      0.24    -0.49
## cor(expected1:phase1,expected1:difficulty1)         0.09      0.23    -0.38
```

```
## cor(expected1:phase2,expected1:difficulty1)          0.01      0.23    -0.44
## cor(phase1:difficulty1,expected1:difficulty1)        0.03      0.23    -0.42
## cor(phase2:difficulty1,expected1:difficulty1)        0.06      0.24    -0.42
## cor(phase1:difficulty2,expected1:difficulty1)        0.03      0.24    -0.44
## cor(phase2:difficulty2,expected1:difficulty1)        0.02      0.24    -0.43
## cor(Intercept,expected1:difficulty2)                -0.01      0.16    -0.32
## cor(expected1,expected1:difficulty2)                 0.25      0.19    -0.13
## cor(phase1,expected1:difficulty2)                    0.39      0.16     0.06
## cor(phase2,expected1:difficulty2)                    0.06      0.19    -0.32
## cor(difficulty1,expected1:difficulty2)               0.00      0.23    -0.44
## cor(difficulty2,expected1:difficulty2)              -0.40      0.20    -0.73
## cor(expected1:phase1,expected1:difficulty2)         -0.00      0.19    -0.37
## cor(expected1:phase2,expected1:difficulty2)         -0.17      0.19    -0.53
## cor(phase1:difficulty1,expected1:difficulty2)       -0.07      0.19    -0.44
## cor(phase2:difficulty1,expected1:difficulty2)       -0.17      0.22    -0.57
## cor(phase1:difficulty2,expected1:difficulty2)        0.12      0.21    -0.30
## cor(phase2:difficulty2,expected1:difficulty2)        0.12      0.21    -0.30
## cor(expected1:difficulty1,expected1:difficulty2)    -0.05      0.24    -0.50
##                                                 u-95% CI Rhat Bulk_ESS
## sd(Intercept)                                       0.27 1.00     2856
## sd(expected1)                                       0.04 1.00     7481
## sd(phase1)                                          0.10 1.00     7310
## sd(phase2)                                          0.05 1.00     7164
## sd(difficulty1)                                     0.03 1.00     4255
## sd(difficulty2)                                     0.03 1.00     5984
## sd(expected1:phase1)                                0.05 1.00     8529
## sd(expected1:phase2)                                0.04 1.00     8720
## sd(phase1:difficulty1)                              0.06 1.00     6990
## sd(phase2:difficulty1)                              0.04 1.00     4295
## sd(phase1:difficulty2)                              0.04 1.00     4189
## sd(phase2:difficulty2)                              0.05 1.00     4041
## sd(expected1:difficulty1)                           0.02 1.00     5474
## sd(expected1:difficulty2)                           0.04 1.00     6138
## cor(Intercept,expected1)                            0.28 1.00    18965
## cor(Intercept,phase1)                               0.32 1.00     8901
## cor(expected1,phase1)                               0.47 1.00     2892
## cor(Intercept,phase2)                               0.12 1.00    18148
## cor(expected1,phase2)                               0.51 1.00     9169
## cor(phase1,phase2)                                 -0.03 1.00    14899
## cor(Intercept,difficulty1)                          0.32 1.00    28933
## cor(expected1,difficulty1)                          0.39 1.00    21720
## cor(phase1,difficulty1)                             0.40 1.00    27327
## cor(phase2,difficulty1)                             0.37 1.00    19557
## cor(Intercept,difficulty2)                          0.17 1.00    24894
## cor(expected1,difficulty2)                          0.16 1.00    13825
## cor(phase1,difficulty2)                             0.08 1.00    21644
## cor(phase2,difficulty2)                             0.32 1.00    16363
## cor(difficulty1,difficulty2)                        0.41 1.00    10344
## cor(Intercept,expected1:phase1)                     0.17 1.00    17506
## cor(expected1,expected1:phase1)                     0.49 1.00     7439
## cor(phase1,expected1:phase1)                       -0.07 1.00    15983
## cor(phase2,expected1:phase1)                        0.41 1.00     9224
## cor(difficulty1,expected1:phase1)                   0.40 1.00     6170
## cor(difficulty2,expected1:phase1)                   0.42 1.00     8101
```

```
## cor(Intercept,expected1:phase2)                       0.64 1.00    22206
## cor(expected1,expected1:phase2)                       0.20 1.00    13022
## cor(phase1,expected1:phase2)                          0.38 1.00    20778
## cor(phase2,expected1:phase2)                         -0.14 1.00    14261
## cor(difficulty1,expected1:phase2)                     0.43 1.00    14604
## cor(difficulty2,expected1:phase2)                     0.49 1.00    16111
## cor(expected1:phase1,expected1:phase2)                0.25 1.00    16972
## cor(Intercept,phase1:difficulty1)                     0.03 1.00    19947
## cor(expected1,phase1:difficulty1)                     0.21 1.00    10070
## cor(phase1,phase1:difficulty1)                        0.39 1.00    17257
## cor(phase2,phase1:difficulty1)                        0.34 1.00    10588
## cor(difficulty1,phase1:difficulty1)                   0.53 1.00     6220
## cor(difficulty2,phase1:difficulty1)                   0.41 1.00     8293
## cor(expected1:phase1,phase1:difficulty1)              0.22 1.00    12805
## cor(expected1:phase2,phase1:difficulty1)              0.25 1.00     9920
## cor(Intercept,phase2:difficulty1)                     0.31 1.00    23181
## cor(expected1,phase2:difficulty1)                     0.23 1.00    16041
## cor(phase1,phase2:difficulty1)                        0.35 1.00    23381
## cor(phase2,phase2:difficulty1)                        0.24 1.00    14377
## cor(difficulty1,phase2:difficulty1)                   0.45 1.00    13649
## cor(difficulty2,phase2:difficulty1)                   0.57 1.00    11815
## cor(expected1:phase1,phase2:difficulty1)              0.45 1.00    19579
## cor(expected1:phase2,phase2:difficulty1)              0.59 1.00    13165
## cor(phase1:difficulty1,phase2:difficulty1)            0.44 1.00    18556
## cor(Intercept,phase1:difficulty2)                     0.46 1.00    23283
## cor(expected1,phase1:difficulty2)                     0.58 1.00    14775
## cor(phase1,phase1:difficulty2)                        0.38 1.00    23858
## cor(phase2,phase1:difficulty2)                        0.58 1.00    13841
## cor(difficulty1,phase1:difficulty2)                   0.36 1.00    10089
## cor(difficulty2,phase1:difficulty2)                   0.28 1.00    13012
## cor(expected1:phase1,phase1:difficulty2)              0.37 1.00    20006
## cor(expected1:phase2,phase1:difficulty2)              0.28 1.00    14034
## cor(phase1:difficulty1,phase1:difficulty2)            0.28 1.00    12571
## cor(phase2:difficulty1,phase1:difficulty2)            0.28 1.00    10328
## cor(Intercept,phase2:difficulty2)                     0.53 1.00    20338
## cor(expected1,phase2:difficulty2)                     0.54 1.00    12487
## cor(phase1,phase2:difficulty2)                        0.29 1.00    22452
## cor(phase2,phase2:difficulty2)                        0.46 1.00    12785
## cor(difficulty1,phase2:difficulty2)                   0.41 1.00    10081
## cor(difficulty2,phase2:difficulty2)                   0.25 1.00     9716
## cor(expected1:phase1,phase2:difficulty2)              0.46 1.00    16835
## cor(expected1:phase2,phase2:difficulty2)              0.43 1.00    13701
## cor(phase1:difficulty1,phase2:difficulty2)            0.19 1.00    14015
## cor(phase2:difficulty1,phase2:difficulty2)            0.29 1.00     8339
## cor(phase1:difficulty2,phase2:difficulty2)            0.54 1.00    11222
## cor(Intercept,expected1:difficulty1)                  0.47 1.00    33288
## cor(expected1,expected1:difficulty1)                  0.49 1.00    26393
## cor(phase1,expected1:difficulty1)                     0.47 1.00    29631
## cor(phase2,expected1:difficulty1)                     0.47 1.00    23764
## cor(difficulty1,expected1:difficulty1)                0.39 1.00     9995
## cor(difficulty2,expected1:difficulty1)                0.44 1.00    20944
## cor(expected1:phase1,expected1:difficulty1)           0.53 1.00    18001
## cor(expected1:phase2,expected1:difficulty1)           0.46 1.00    23210
## cor(phase1:difficulty1,expected1:difficulty1)         0.46 1.00    21030
```

```
## cor(phase2:difficulty1,expected1:difficulty1)        0.51 1.00    14379
## cor(phase1:difficulty2,expected1:difficulty1)        0.48 1.00    17098
## cor(phase2:difficulty2,expected1:difficulty1)        0.47 1.00    18923
## cor(Intercept,expected1:difficulty2)                 0.30 1.00    28157
## cor(expected1,expected1:difficulty2)                 0.59 1.00    10983
## cor(phase1,expected1:difficulty2)                    0.67 1.00    16883
## cor(phase2,expected1:difficulty2)                    0.42 1.00    15766
## cor(difficulty1,expected1:difficulty2)               0.44 1.00     9749
## cor(difficulty2,expected1:difficulty2)               0.04 1.00     7834
## cor(expected1:phase1,expected1:difficulty2)          0.37 1.00    16236
## cor(expected1:phase2,expected1:difficulty2)          0.22 1.00    14644
## cor(phase1:difficulty1,expected1:difficulty2)        0.32 1.00    18071
## cor(phase2:difficulty1,expected1:difficulty2)        0.28 1.00    11930
## cor(phase1:difficulty2,expected1:difficulty2)        0.52 1.00    15095
## cor(phase2:difficulty2,expected1:difficulty2)        0.51 1.00    15050
## cor(expected1:difficulty1,expected1:difficulty2)     0.42 1.00    13178
##                                                  Tail_ESS
## sd(Intercept)                                        5862
## sd(expected1)                                        9174
## sd(phase1)                                          12662
## sd(phase2)                                           9347
## sd(difficulty1)                                      7392
## sd(difficulty2)                                      6047
## sd(expected1:phase1)                                 9238
## sd(expected1:phase2)                                 9521
## sd(phase1:difficulty1)                               7279
## sd(phase2:difficulty1)                               5112
## sd(phase1:difficulty2)                               4702
## sd(phase2:difficulty2)                               2793
## sd(expected1:difficulty1)                            8408
## sd(expected1:difficulty2)                            4798
## cor(Intercept,expected1)                            14369
## cor(Intercept,phase1)                               12476
## cor(expected1,phase1)                                5556
## cor(Intercept,phase2)                               14623
## cor(expected1,phase2)                               12169
## cor(phase1,phase2)                                  13901
## cor(Intercept,difficulty1)                          11981
## cor(expected1,difficulty1)                          13268
## cor(phase1,difficulty1)                             13644
## cor(phase2,difficulty1)                             13471
## cor(Intercept,difficulty2)                          14508
## cor(expected1,difficulty2)                          14526
## cor(phase1,difficulty2)                             13387
## cor(phase2,difficulty2)                             14306
## cor(difficulty1,difficulty2)                        14749
## cor(Intercept,expected1:phase1)                     14717
## cor(expected1,expected1:phase1)                     11721
## cor(phase1,expected1:phase1)                        15226
## cor(phase2,expected1:phase1)                        13146
## cor(difficulty1,expected1:phase1)                    9708
## cor(difficulty2,expected1:phase1)                   11967
## cor(Intercept,expected1:phase2)                     13851
## cor(expected1,expected1:phase2)                     13747
```

```
## cor(phase1,expected1:phase2)                          14317
## cor(phase2,expected1:phase2)                          12968
## cor(difficulty1,expected1:phase2)                     14199
## cor(difficulty2,expected1:phase2)                     15432
## cor(expected1:phase1,expected1:phase2)                14828
## cor(Intercept,phase1:difficulty1)                     14040
## cor(expected1,phase1:difficulty1)                     13409
## cor(phase1,phase1:difficulty1)                        14670
## cor(phase2,phase1:difficulty1)                        14189
## cor(difficulty1,phase1:difficulty1)                   10529
## cor(difficulty2,phase1:difficulty1)                   11800
## cor(expected1:phase1,phase1:difficulty1)              15086
## cor(expected1:phase2,phase1:difficulty1)              14532
## cor(Intercept,phase2:difficulty1)                     13262
## cor(expected1,phase2:difficulty1)                     13963
## cor(phase1,phase2:difficulty1)                        14399
## cor(phase2,phase2:difficulty1)                        12482
## cor(difficulty1,phase2:difficulty1)                   13883
## cor(difficulty2,phase2:difficulty1)                   13606
## cor(expected1:phase1,phase2:difficulty1)              14444
## cor(expected1:phase2,phase2:difficulty1)              13734
## cor(phase1:difficulty1,phase2:difficulty1)            15247
## cor(Intercept,phase1:difficulty2)                     13194
## cor(expected1,phase1:difficulty2)                     13909
## cor(phase1,phase1:difficulty2)                        13647
## cor(phase2,phase1:difficulty2)                        13263
## cor(difficulty1,phase1:difficulty2)                   12879
## cor(difficulty2,phase1:difficulty2)                   13941
## cor(expected1:phase1,phase1:difficulty2)              15500
## cor(expected1:phase2,phase1:difficulty2)              14711
## cor(phase1:difficulty1,phase1:difficulty2)            13441
## cor(phase2:difficulty1,phase1:difficulty2)            13628
## cor(Intercept,phase2:difficulty2)                     12729
## cor(expected1,phase2:difficulty2)                     13198
## cor(phase1,phase2:difficulty2)                        14680
## cor(phase2,phase2:difficulty2)                        14117
## cor(difficulty1,phase2:difficulty2)                   12425
## cor(difficulty2,phase2:difficulty2)                   13542
## cor(expected1:phase1,phase2:difficulty2)              15511
## cor(expected1:phase2,phase2:difficulty2)              14597
## cor(phase1:difficulty1,phase2:difficulty2)            14144
## cor(phase2:difficulty1,phase2:difficulty2)            13046
## cor(phase1:difficulty2,phase2:difficulty2)            12661
## cor(Intercept,expected1:difficulty1)                  12937
## cor(expected1,expected1:difficulty1)                  13793
## cor(phase1,expected1:difficulty1)                     13501
## cor(phase2,expected1:difficulty1)                     14402
## cor(difficulty1,expected1:difficulty1)                13654
## cor(difficulty2,expected1:difficulty1)                13893
## cor(expected1:phase1,expected1:difficulty1)           14417
## cor(expected1:phase2,expected1:difficulty1)           15342
## cor(phase1:difficulty1,expected1:difficulty1)         14581
## cor(phase2:difficulty1,expected1:difficulty1)         15251
## cor(phase1:difficulty2,expected1:difficulty1)         15282
```

```
## cor(phase2:difficulty2,expected1:difficulty1)       15692
## cor(Intercept,expected1:difficulty2)                14540
## cor(expected1,expected1:difficulty2)                13160
## cor(phase1,expected1:difficulty2)                   12322
## cor(phase2,expected1:difficulty2)                   15623
## cor(difficulty1,expected1:difficulty2)              13289
## cor(difficulty2,expected1:difficulty2)               9666
## cor(expected1:phase1,expected1:difficulty2)         15347
## cor(expected1:phase2,expected1:difficulty2)         13483
## cor(phase1:difficulty1,expected1:difficulty2)       16018
## cor(phase2:difficulty1,expected1:difficulty2)       13293
## cor(phase1:difficulty2,expected1:difficulty2)       15431
## cor(phase2:difficulty2,expected1:difficulty2)       13954
## cor(expected1:difficulty1,expected1:difficulty2)    15185
##
## Regression Coefficients:
##                                      Estimate Est.Error l-95% CI u-95% CI
## Intercept                                5.91      0.04     5.83     5.98
## diagnosis1                               0.02      0.03    -0.03     0.07
## diagnosis2                               0.03      0.03    -0.02     0.09
## diagnosis3                               0.02      0.03    -0.04     0.07
## 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      0.00    -0.05    -0.03
## difficulty2                             -0.01      0.00    -0.02    -0.00
## diagnosis1:expected1                    -0.01      0.01    -0.02     0.01
## diagnosis2:expected1                     0.01      0.01    -0.00     0.02
## diagnosis3:expected1                    -0.00      0.01    -0.02     0.01
## diagnosis1:phase1                       -0.02      0.02    -0.05     0.01
## diagnosis2:phase1                        0.01      0.02    -0.02     0.04
## diagnosis3:phase1                       -0.01      0.01    -0.04     0.02
## diagnosis1:phase2                       -0.00      0.01    -0.02     0.02
## diagnosis2:phase2                        0.00      0.01    -0.02     0.02
## diagnosis3:phase2                        0.00      0.01    -0.01     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    -0.04    -0.00
## diagnosis2:difficulty1                   0.01      0.01    -0.00     0.03
## diagnosis3:difficulty1                   0.01      0.01    -0.01     0.02
## diagnosis1:difficulty2                   0.02      0.01     0.01     0.04
## diagnosis2:difficulty2                  -0.01      0.01    -0.03     0.01
## diagnosis3:difficulty2                  -0.02      0.01    -0.04    -0.00
## expected1:difficulty1                   -0.02      0.00    -0.03    -0.01
## expected1:difficulty2                   -0.00      0.01    -0.01     0.01
## phase1:difficulty1                      -0.00      0.01    -0.02     0.01
## phase2:difficulty1                       0.00      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    -0.02     0.02
## diagnosis2:expected1:phase1              0.01      0.01    -0.01     0.03
## diagnosis3:expected1:phase1              0.01      0.01    -0.01     0.02
## diagnosis1:expected1:phase2              0.00      0.01    -0.02     0.02
## diagnosis2:expected1:phase2             -0.01      0.01    -0.02     0.01
```

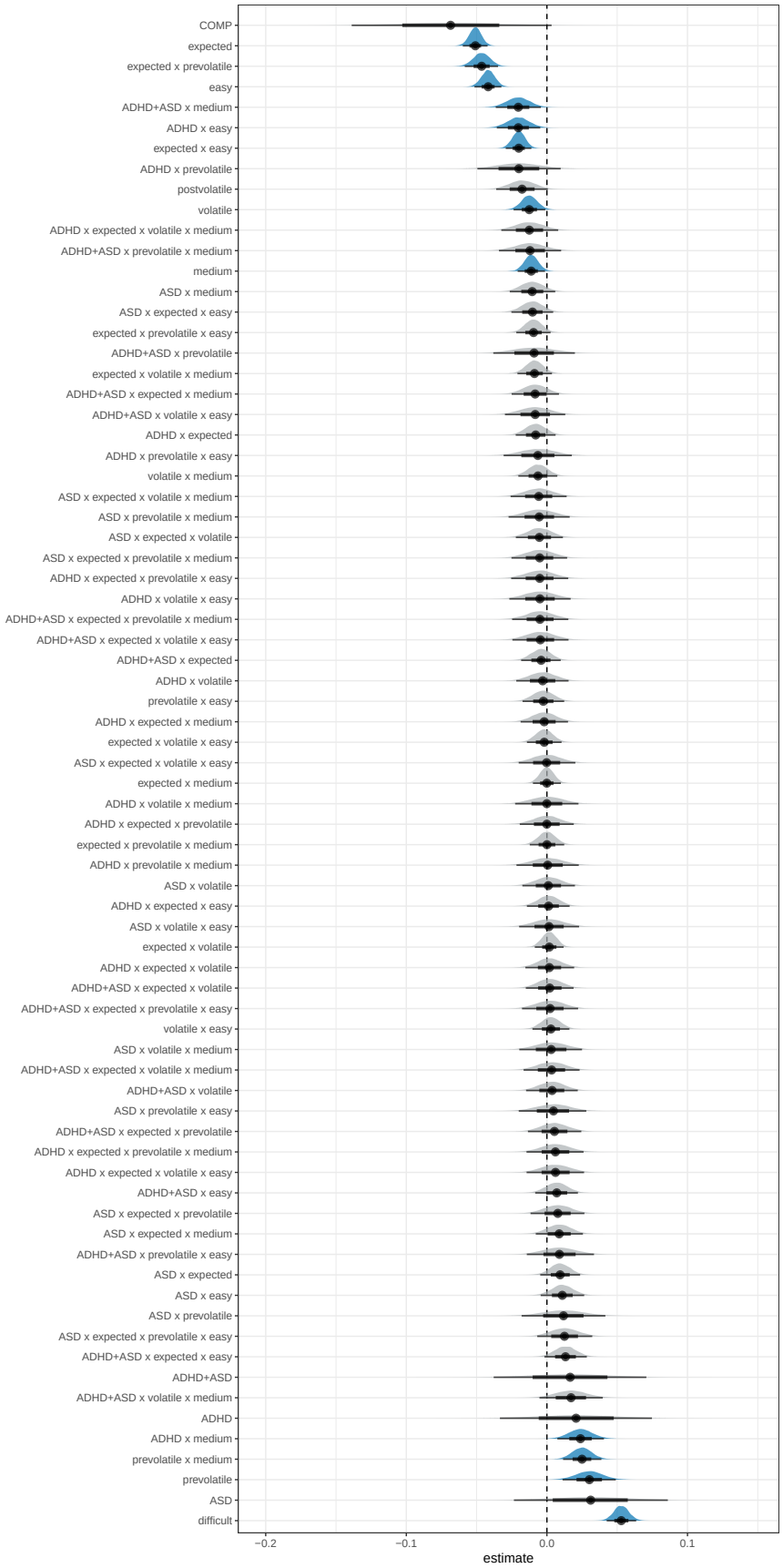
```
## diagnosis3:expected1:phase2                0.00      0.01    -0.02     0.02
## diagnosis1:expected1:difficulty1           0.00      0.01    -0.01     0.02
## diagnosis2:expected1:difficulty1          -0.01      0.01    -0.03     0.00
## diagnosis3:expected1:difficulty1           0.01      0.01    -0.00     0.03
## diagnosis1:expected1:difficulty2          -0.00      0.01    -0.02     0.02
## diagnosis2:expected1:difficulty2           0.01      0.01    -0.01     0.03
## diagnosis3:expected1:difficulty2          -0.01      0.01    -0.03     0.01
## diagnosis1:phase1:difficulty1             -0.01      0.01    -0.03     0.02
## diagnosis2:phase1:difficulty1              0.00      0.01    -0.02     0.03
## diagnosis3:phase1:difficulty1              0.01      0.01    -0.01     0.03
## diagnosis1:phase2:difficulty1             -0.00      0.01    -0.03     0.02
## diagnosis2:phase2:difficulty1              0.00      0.01    -0.02     0.02
## diagnosis3:phase2:difficulty1             -0.01      0.01    -0.03     0.01
## diagnosis1:phase1:difficulty2              0.00      0.01    -0.02     0.02
## diagnosis2:phase1:difficulty2             -0.01      0.01    -0.03     0.02
## diagnosis3:phase1:difficulty2             -0.01      0.01    -0.03     0.01
## diagnosis1:phase2:difficulty2             -0.00      0.01    -0.02     0.02
## diagnosis2:phase2:difficulty2              0.00      0.01    -0.02     0.02
## diagnosis3:phase2:difficulty2              0.02      0.01    -0.01     0.04
## 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     0.01
## expected1:phase2:difficulty2              -0.01      0.01    -0.02     0.00
## diagnosis1:expected1:phase1:difficulty1   -0.01      0.01    -0.03     0.02
## diagnosis2:expected1:phase1:difficulty1    0.01      0.01    -0.01     0.03
## diagnosis3:expected1:phase1:difficulty1    0.00      0.01    -0.02     0.02
## diagnosis1:expected1:phase2:difficulty1    0.01      0.01    -0.01     0.03
## diagnosis2:expected1:phase2:difficulty1   -0.00      0.01    -0.02     0.02
## diagnosis3:expected1:phase2:difficulty1   -0.00      0.01    -0.02     0.02
## diagnosis1:expected1:phase1:difficulty2    0.01      0.01    -0.01     0.03
## diagnosis2:expected1:phase1:difficulty2   -0.01      0.01    -0.03     0.01
## diagnosis3:expected1:phase1:difficulty2   -0.00      0.01    -0.02     0.02
## diagnosis1:expected1:phase2:difficulty2   -0.01      0.01    -0.03     0.01
## diagnosis2:expected1:phase2:difficulty2   -0.01      0.01    -0.03     0.01
## diagnosis3:expected1:phase2:difficulty2    0.00      0.01    -0.02     0.02
##                                         Rhat Bulk_ESS Tail_ESS
## Intercept                               1.00     2661     5819
## diagnosis1                              1.00     2930     5477
## diagnosis2                              1.00     2431     4786
## diagnosis3                              1.00     2670     5263
## expected1                               1.00    14025    14622
## phase1                                  1.00     7141    10612
## phase2                                  1.00    15633    15040
## difficulty1                             1.00    23341    15599
## difficulty2                             1.00    23067    14517
## diagnosis1:expected1                    1.00    14562    14155
## diagnosis2:expected1                    1.00    14452    13537
## diagnosis3:expected1                    1.00    15072    13967
## diagnosis1:phase1                       1.00     7948    10949
## diagnosis2:phase1                       1.00     8075    11199
## diagnosis3:phase1                       1.00     8335    11401
## diagnosis1:phase2                       1.00    15944    13669
## diagnosis2:phase2                       1.00    15022    14334
## diagnosis3:phase2                       1.00    14754    14425
```

```
## expected1:phase1                          1.00    15276    14970
## expected1:phase2                          1.00    14638    13666
## diagnosis1:difficulty1                    1.00    22542    14994
## diagnosis2:difficulty1                    1.00    23362    15115
## diagnosis3:difficulty1                    1.00    22341    14746
## diagnosis1:difficulty2                    1.00    19671    15529
## diagnosis2:difficulty2                    1.00    20317    14488
## diagnosis3:difficulty2                    1.00    19230    14511
## expected1:difficulty1                     1.00    28914    14887
## expected1:difficulty2                     1.00    22237    15425
## phase1:difficulty1                        1.00    15198    15367
## phase2:difficulty1                        1.00    22344    14326
## phase1:difficulty2                        1.00    20487    14161
## phase2:difficulty2                        1.00    16877    14766
## diagnosis1:expected1:phase1               1.00    15118    14695
## diagnosis2:expected1:phase1               1.00    15370    14617
## diagnosis3:expected1:phase1               1.00    14730    14149
## diagnosis1:expected1:phase2               1.00    19119    15070
## diagnosis2:expected1:phase2               1.00    18142    14652
## diagnosis3:expected1:phase2               1.00    16940    13964
## diagnosis1:expected1:difficulty1          1.00    23155    15466
## diagnosis2:expected1:difficulty1          1.00    22542    14479
## diagnosis3:expected1:difficulty1          1.00    21847    14532
## diagnosis1:expected1:difficulty2          1.00    18001    15211
## diagnosis2:expected1:difficulty2          1.00    17993    14809
## diagnosis3:expected1:difficulty2          1.00    18446    14712
## diagnosis1:phase1:difficulty1             1.00    16157    13405
## diagnosis2:phase1:difficulty1             1.00    15439    14860
## diagnosis3:phase1:difficulty1             1.00    16619    14940
## diagnosis1:phase2:difficulty1             1.00    18120    14244
## diagnosis2:phase2:difficulty1             1.00    18797    14850
## diagnosis3:phase2:difficulty1             1.00    18357    14998
## diagnosis1:phase1:difficulty2             1.00    19398    15122
## diagnosis2:phase1:difficulty2             1.00    20290    15188
## diagnosis3:phase1:difficulty2             1.00    18918    14982
## diagnosis1:phase2:difficulty2             1.00    17264    14384
## diagnosis2:phase2:difficulty2             1.00    17973    14267
## diagnosis3:phase2:difficulty2             1.00    17406    15250
## expected1:phase1:difficulty1              1.00    24306    14342
## expected1:phase2:difficulty1              1.00    24116    14787
## expected1:phase1:difficulty2              1.00    25324    14184
## expected1:phase2:difficulty2              1.00    24141    15208
## diagnosis1:expected1:phase1:difficulty1 1.00    19924    14996
## diagnosis2:expected1:phase1:difficulty1 1.00    19241    15105
## diagnosis3:expected1:phase1:difficulty1 1.00    19096    15632
## diagnosis1:expected1:phase2:difficulty1 1.00    19299    14432
## diagnosis2:expected1:phase2:difficulty1 1.00    19473    14916
## diagnosis3:expected1:phase2:difficulty1 1.00    19048    15284
## diagnosis1:expected1:phase1:difficulty2 1.00    19780    14956
## diagnosis2:expected1:phase1:difficulty2 1.00    19700    15546
## diagnosis3:expected1:phase1:difficulty2 1.00    19721    14962
## diagnosis1:expected1:phase2:difficulty2 1.00    19234    14041
## diagnosis2:expected1:phase2:difficulty2 1.00    18734    14459
## diagnosis3:expected1:phase2:difficulty2 1.00    18292    14897
```

```
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.13      0.01     0.12     0.14 1.00     5483    10330
## ndt     216.55     10.38   194.46   234.99 1.00     9267    11437
##
## 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_diagnosis3,
    b_postvolatile = - b_phase1 - b_phase2,
    b_difficult    = - b_difficulty1 - b_difficulty2,
    ASD            = b_Intercept + b_diagnosis2,
    BOTH           = b_Intercept + b_diagnosis3,
    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`,
    BOTH_expected   = b_Intercept + b_diagnosis3 + b_expected1 +
      `b_diagnosis3:expected1`,
    BOTH_unexpected = b_Intercept + b_diagnosis3 - b_expected1 -
      `b_diagnosis3:expected1`,
    COMP_expected   = b_Intercept - b_diagnosis1 - b_diagnosis2 - b_diagnosis3 + b_expected1 -
      `b_diagnosis1:expected1` - `b_diagnosis2:expected1` - `b_diagnosis3:expected1`,
    COMP_unexpected   = b_Intercept - b_diagnosis1 - b_diagnosis2 - b_diagnosis3 -
      b_expected1 + `b_diagnosis1:expected1` + `b_diagnosis2:expected1` - `b_diagnosis3: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, "diagnosis3", "ADHD+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
ADHD+ASD x medium
ADHD x easy
expected x easy
ADHD x prevolatile
postvolatile
volatile
ADHD x expected x volatile x medium
ADHD+ASD x prevolatile x medium
medium
ASD x medium
ASD x expected x easy
expected x prevolatile x easy
ADHD+ASD x prevolatile
expected x volatile x medium
ADHD+ASD x expected x medium
ADHD+ASD x volatile x easy
ADHD x expected
ADHD x prevolatile x easy
volatile x medium
ASD x expected x volatile x medium
ASD x prevolatile x medium
ASD x expected x volatile
ASD x expected x prevolatile x medium
ADHD x expected x prevolatile x easy
ADHD x volatile x easy
ADHD+ASD x expected x prevolatile x medium
ADHD+ASD x expected x volatile x easy
ADHD+ASD x expected
ADHD x volatile
prevolatile x easy
ADHD x expected x medium
expected x volatile x easy
ASD x expected x volatile x easy
expected x medium
ADHD x volatile x medium
ADHD x expected x prevolatile
expected x prevolatile x medium
ADHD x prevolatile x medium
ASD x volatile
ADHD x expected x easy
ASD x volatile x easy
expected x volatile
ADHD x expected x volatile
ADHD+ASD x expected x volatile
ADHD+ASD x expected x prevolatile x easy
volatile x easy
ASD x volatile x medium
ADHD+ASD x expected x volatile x medium
ADHD+ASD x volatile
ASD x prevolatile x easy
ADHD+ASD x expected x prevolatile
ADHD x expected x prevolatile x medium
ADHD x expected x volatile x easy
ADHD+ASD x easy
ASD x expected x prevolatile
ASD x expected x medium
ADHD+ASD x prevolatile x easy
ASD x expected
ASD x easy
ASD x prevolatile
ASD x expected x prevolatile x easy
ADHD+ASD x expected x easy
ADHD+ASD
ADHD+ASD x volatile x medium
ADHD
ADHD x medium
prevolatile x medium
prevolatile
ASD
difficult
−0.2
−0.1
0.0
0.1
estimate

```
## [!CHECK THOSE AGAIN!!!!

# H1b: COMP(unexp-exp) > ASD(unexp-exp)
h1b = hypothesis(m.pal, "0 <
                 2*(diagnosis1:expected1 + 2*diagnosis2:expected1+ diagnosis3:expected1)")
h1b
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*(diagnosis... < 0    -0.01      0.02    -0.05     0.03       2.52
##   Post.Prob Star
## 1      0.72
## ---
## '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*(2*diagnosis1:expected1 + diagnosis2:expected1+ diagnosis3:expected1)",
                 alpha = 0.025)
h1c
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*(2*diagnos... > 0     0.02      0.02    -0.03     0.07       4.43
##   Post.Prob Star
## 1      0.82
## ---
## '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, "diagnosis1:phase1 + diagnosis2:phase1 + diagnosis3:phase1 -
                 diagnosis1:phase2 - diagnosis2:phase2 - diagnosis3:phase2 <
                 -diagnosis2:phase1 + diagnosis2:phase2")
h1d
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (diagnosis1:phase... < 0    -0.01      0.03    -0.07     0.05       1.46
##   Post.Prob Star
## 1      0.59
## ---
## '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, 0.08]
## h1c       | Accepted |    100.00 % | [-0.07, 0.04]
## h1d       | Accepted |    100.00 % | [-0.04, 0.04]
```

```
## exploration: prevolatile phase

# prevolatile: COMP(unexpected - expected) != ASD(unexpected - expected)
e1.1 = hypothesis(m.pal, "0 < 2*(diagnosis1:expected1 + 2*diagnosis2:expected1 + diagnosis3:expected1 +
                  diagnosis1:expected1:phase1 + 2*diagnosis2:expected1:phase1 + diagnosis3:expected1:pha
                  alpha = 0.025)
e1.1
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*(diagnosis... < 0    -0.05      0.04    -0.14     0.03       9.27
##   Post.Prob Star
## 1       0.9
## ---
## '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*(2*diagnosis1:expected1 + diagnosis2:expected1 + diagnosis3:expected1 +
                  2*diagnosis1:expected1:phase1 + diagnosis2:expected1:phase1 + diagnosis3:expected1:pha
e1.2
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*(2*diagnos... < 0        0      0.04    -0.09     0.08       1.18
##   Post.Prob Star
## 1      0.54
## ---
## '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) != BOTH(unexpected - expected)
e1.3 = hypothesis(m.pal, "0 < 2*(diagnosis1:expected1 + diagnosis2:expected1 + 2*diagnosis3:expected1 +
                  diagnosis1:expected1:phase1 + diagnosis2:expected1:phase1 + 2*diagnosis3:expected1:pha
e1.3
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*(diagnosis... < 0    -0.02      0.04     -0.1     0.06       2.44
##   Post.Prob Star
## 1      0.71
## ---
## '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 > 2*expected1", alpha = 0.025)
e2.1
```

```
## Hypothesis Tests for class b:
##              Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*expected1) > 0      0.1      0.01     0.08     0.12        Inf
##   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.
```

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

```
## 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       2999
##   Post.Prob Star
## 1         1    *
## ---
## 'CI': 95%-CI for one-sided and 97.5%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 97.5%;
## for two-sided hypotheses, the value tested against lies outside the 97.5%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

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

```
## 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       1.99
##   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.
```

```
# difficult versus medium
e2.4 = hypothesis(m.pal, "0 < -difficulty1 - 2*difficulty2", alpha = 0.025)
e2.4
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-difficulty1... < 0    -0.06      0.01    -0.08    -0.05        Inf
```

```
##   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.
```

```
# medium versus easy
e2.5 = hypothesis(m.pal, "0 < -difficulty1 + difficulty2", alpha = 0.025)
e2.5
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-difficulty1... < 0    -0.03      0.01    -0.05    -0.01      17999
##   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.
```

```
# difficult versus easy
e2.6 = hypothesis(m.pal, "0 < -2*difficulty1 - difficulty2", alpha = 0.025)
e2.6
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-2*difficult... < 0    -0.09      0.01    -0.11    -0.08        Inf
##   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_.*"))),
    BOTH_pre_unexp    = rowMeans(across(matches("BOTH_prevolatile_unexpected_.*"))),
    BOTH_pre_exp      = rowMeans(across(matches("BOTH_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_COMPvBOTH_exp =
      (COMP_pre_unexp - COMP_pre_exp) - (BOTH_pre_unexp - BOTH_pre_exp),
    e21_exp           = rowMeans(across(matches(".*_unexpected_.*"))) -
      rowMeans(across(matches(".*_expected_.*"))),
    e22_volvpre       = rowMeans(across(matches(".*_volatile_.*"))) -
      rowMeans(across(matches(".*_prevolatile_.*"))),
    e23_volvpost      = rowMeans(across(matches(".*_volatile_.*"))) -
      rowMeans(across(matches(".*_postvolatile_.*"))),
    e24_diffvmed      = rowMeans(across(matches(".*_difficult"))) -
      rowMeans(across(matches(".*_medium"))),
    e25_medveasy      = rowMeans(across(matches(".*_medium"))) -
      rowMeans(across(matches(".*_easy"))),
    e26_diffveasy     = rowMeans(across(matches(".*_difficult"))) -
      rowMeans(across(matches(".*_easy")))
  )

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

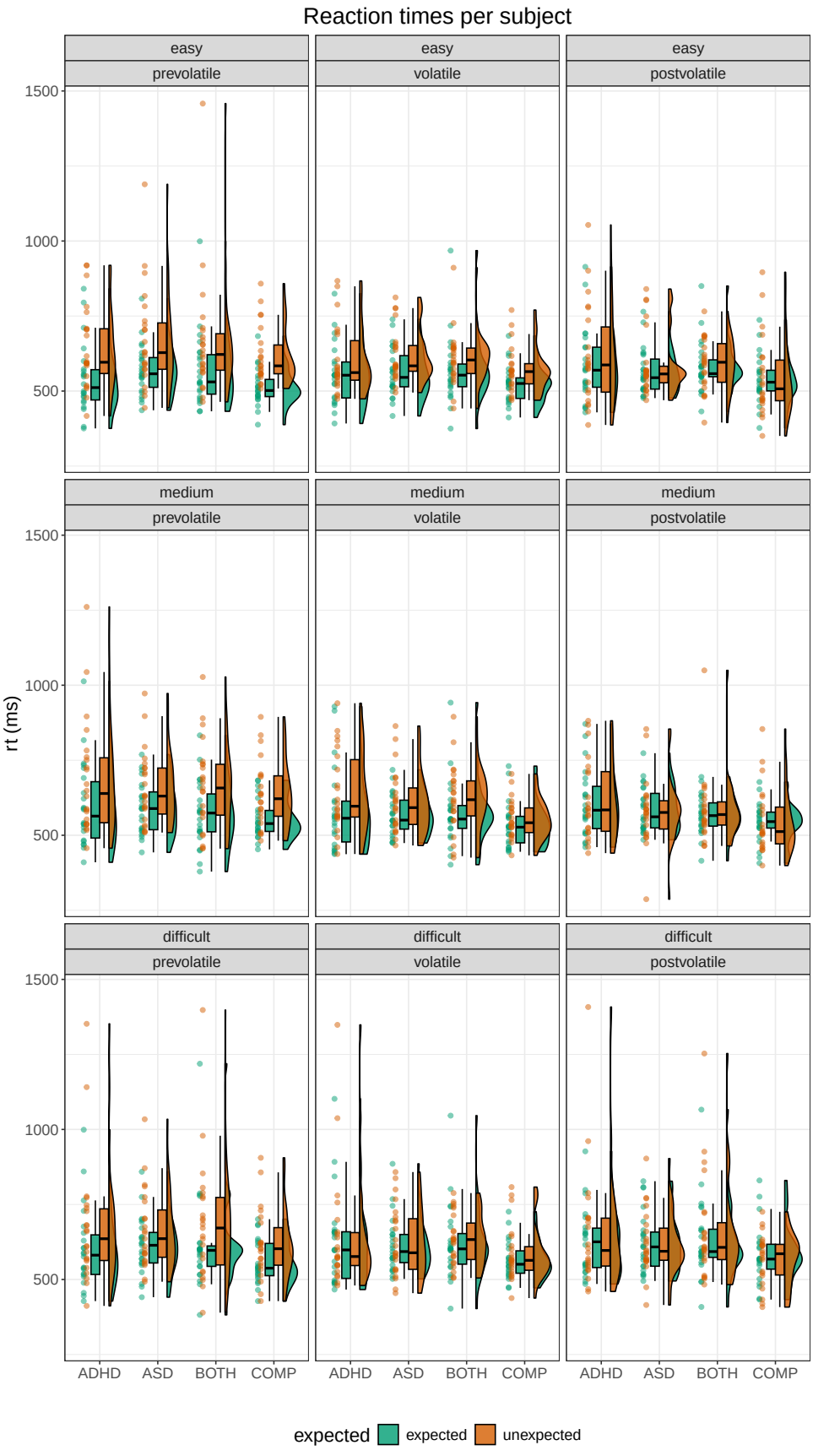
```
## $e11_pre_COMPvASD_exp
## 95% ETI: [-19.08, 46.74]
##
## $e12_pre_COMPvADHD_exp
## 95% ETI: [-35.02, 30.99]
##
## $e13_pre_COMPvBOTH_exp
## 95% ETI: [-28.58, 36.46]
##
```

```
## $e21_exp
## 95% ETI: [31.59, 44.52]
##
## $e22_volvpre
## 95% ETI: [-27.14, -7.61]
##
## $e23_volvpost
## 95% ETI: [-6.72, 10.79]
##
## $e24_diffvmed
## 95% ETI: [16.70, 30.39]
##
## $e25_medveasy
## 95% ETI: [5.00, 16.72]
##
## $e26_diffveasy
## 95% ETI: [27.79, 41.06]
##
## $h1b_COMPvASD_exp
## 95% ETI: [-15.23, 19.30]
##
## $h1c_COMPvADHD_exp
## 95% ETI: [-27.76, 7.49]
##
## $h1d_ASDvCOMP_volvpre
## 95% ETI: [-23.96, 28.14]
```

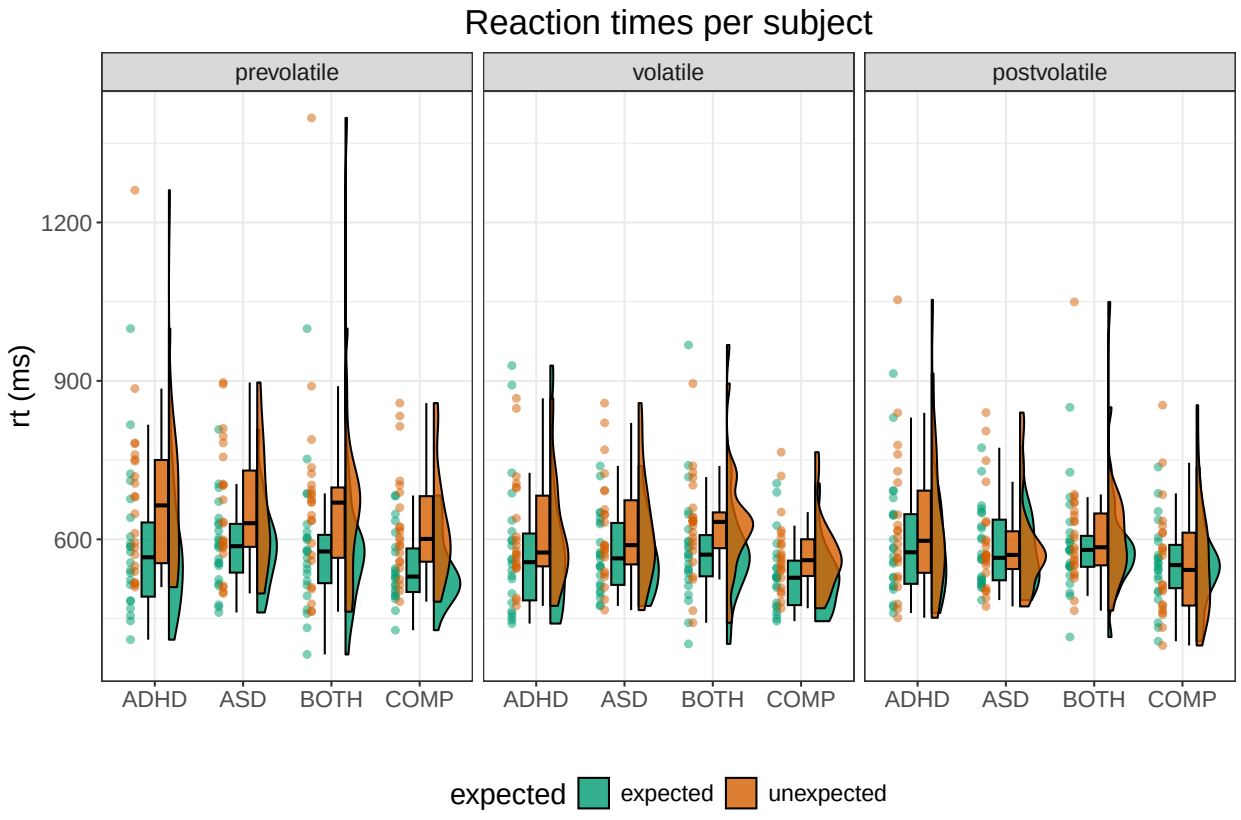
[!MISSING]

Plots

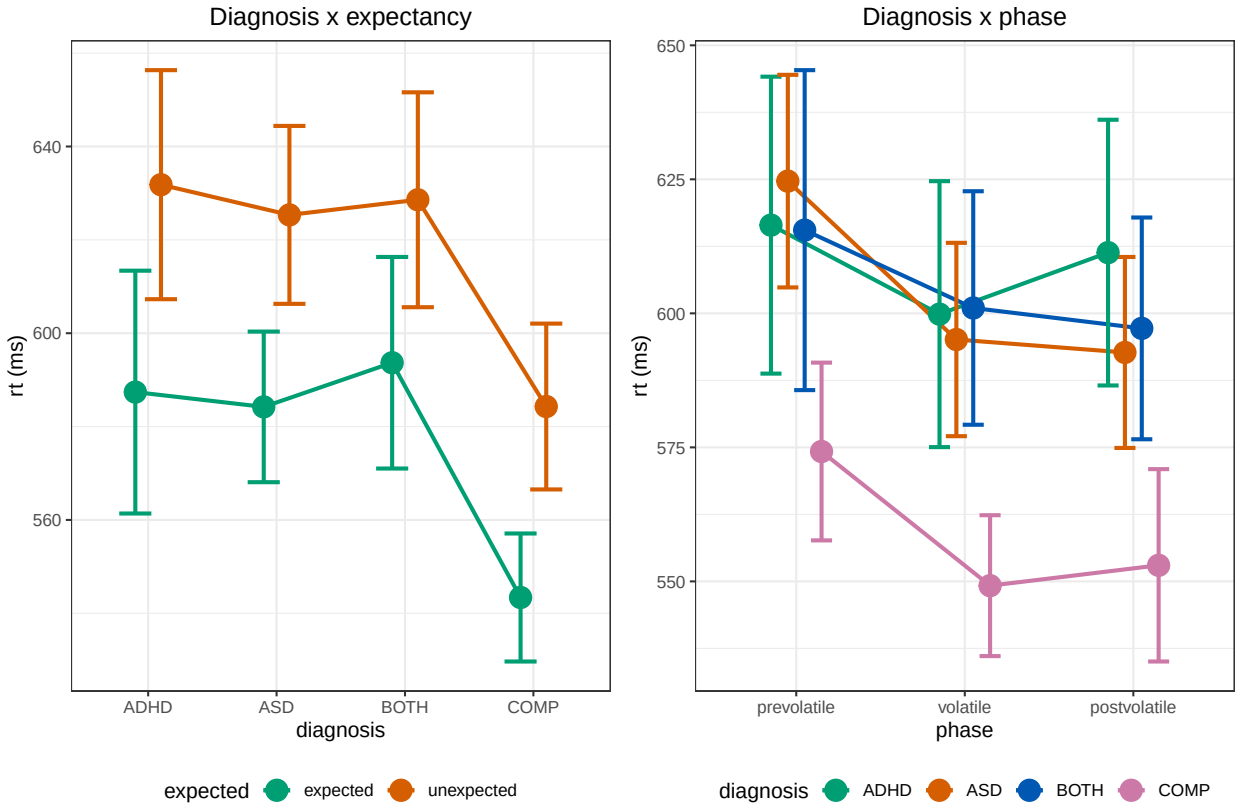
```
# rain cloud plot including all factors

df.pal %>%
  ggplot(aes(diagnosis, rt.cor, fill = expected, colour = expected)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show.legend = FALSE, alpha = .8),
violin.args = list(color = "black", outlier.shape = NA, alpha = .8),
boxplot.args.pos = list(
  position = ggpp::position_dodgenudge(x = 0, width = 0.3), width = 0.3
),
point.args = list(show_guide = FALSE, alpha = .5),
violin.args.pos = list(
  width = 0.6, position = position_nudge(x = 0.16)),
point.args.pos = list(position = ggpp::position_dodgenudge(x = -0.25, width = 0.1))) +
  scale_fill_manual(values = custom.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)
1500
1000
500
ADHD
ASD
BOTH
COMP
expected
expected
unexpected

```
# rain cloud plot excluding difficulty

df.pal %>%
  group_by(subID, diagnosis, expected, phase) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  ) %>%
  ggplot(aes(diagnosis, rt.cor, fill = expected, colour = expected)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show.legend = FALSE, alpha = .8),
violin.args = list(color = "black", outlier.shape = NA, alpha = .8),
boxplot.args.pos = list(
  position = ggpp::position_dodgenudge(x = 0, width = 0.3), width = 0.3
),
point.args = list(show_guide = FALSE, alpha = .5),
violin.args.pos = list(
  width = 0.6, position = position_nudge(x = 0.16)),
point.args.pos = list(position = ggpp::position_dodgenudge(x = -0.25, width = 0.1))) +
  scale_fill_manual(values = custom.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))
```

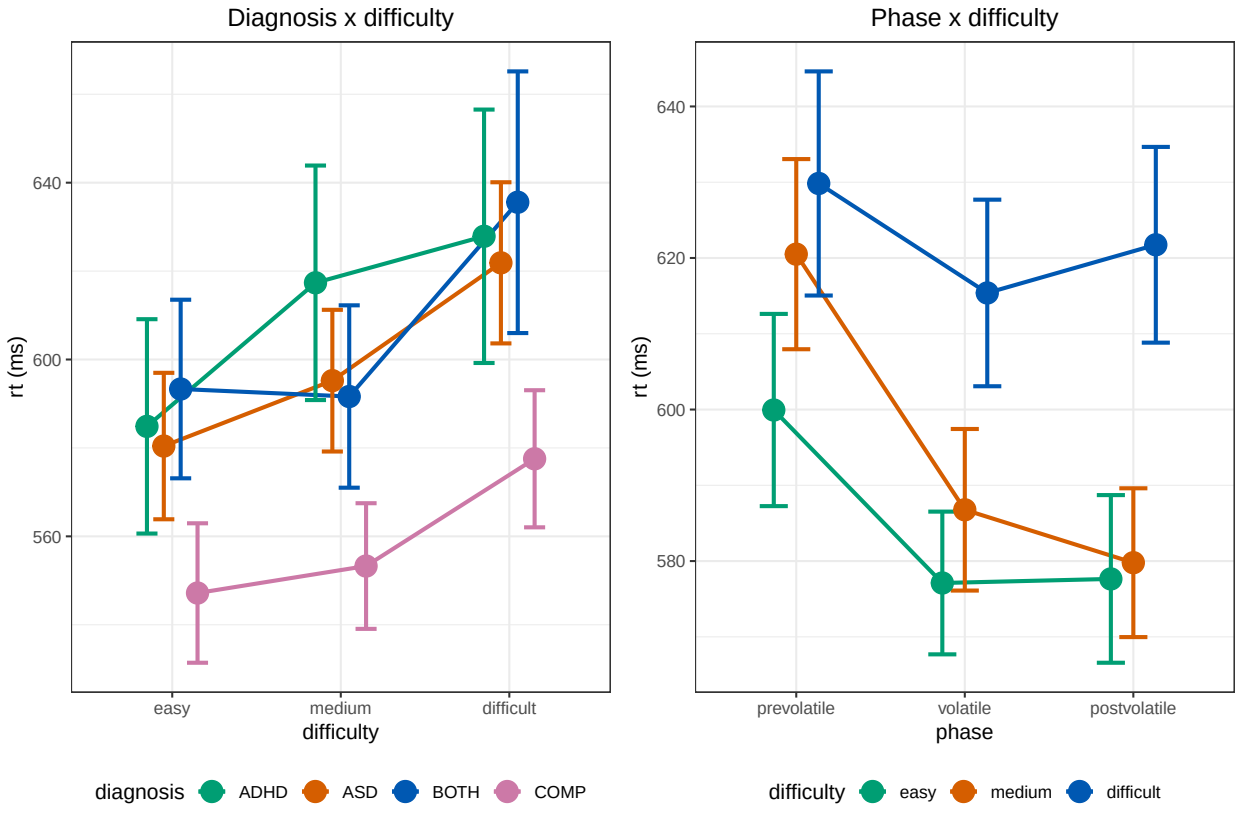


```r
# two-way interactions
p1 = df.pal %>%
  group_by(subID, diagnosis, expected) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  ) %>%
  group_by(diagnosis, expected) %>%
  summarise(
    rt.mn = mean(rt.cor),
    rt.se = sd(rt.cor) / sqrt(n())
  ) %>%
  ggplot(aes(y = rt.mn, x = diagnosis, 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)
```



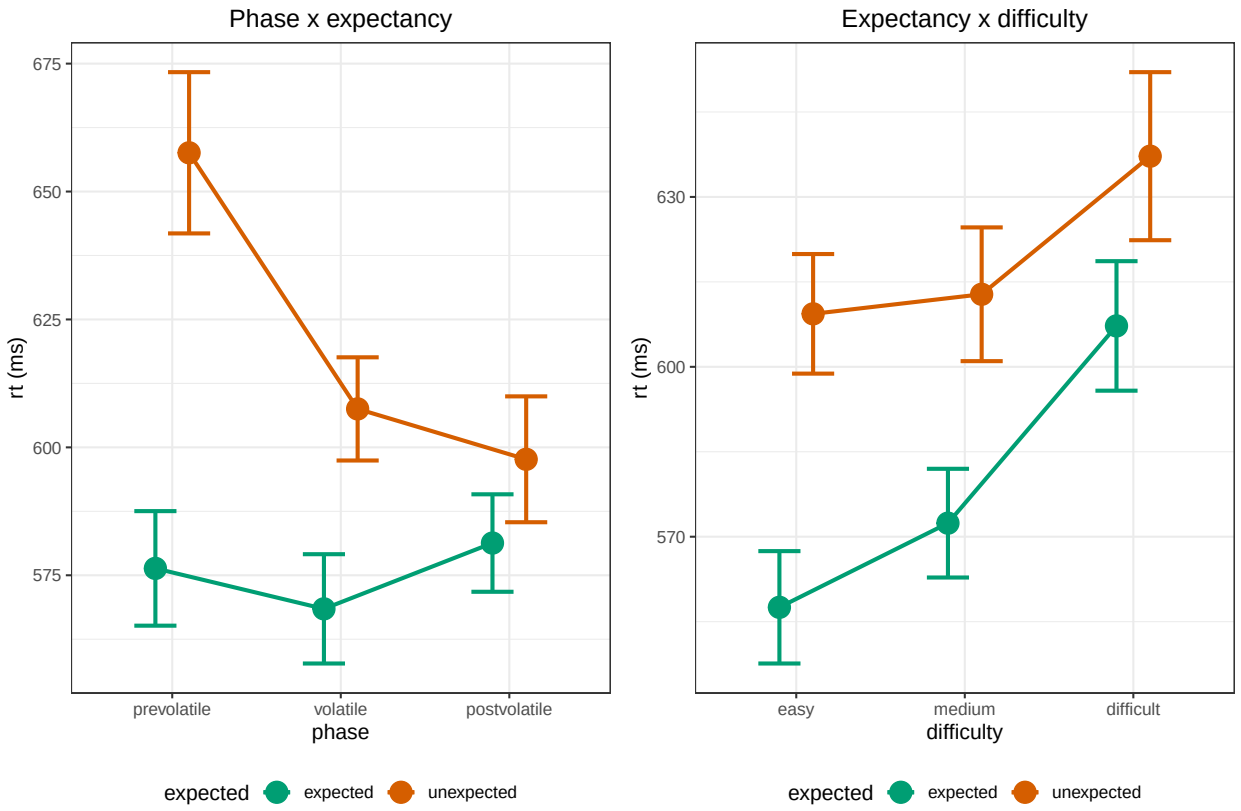

```
p3 = df.pal %>%
  group_by(subID, diagnosis, difficulty) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  )  %>%
  group_by(diagnosis, difficulty) %>%
  summarise(
    rt.mn = mean(rt.cor),
    rt.se = sd(rt.cor) / sqrt(n())
  ) %>%
  ggplot(aes(y = rt.mn, x = difficulty, group = diagnosis, 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)
```

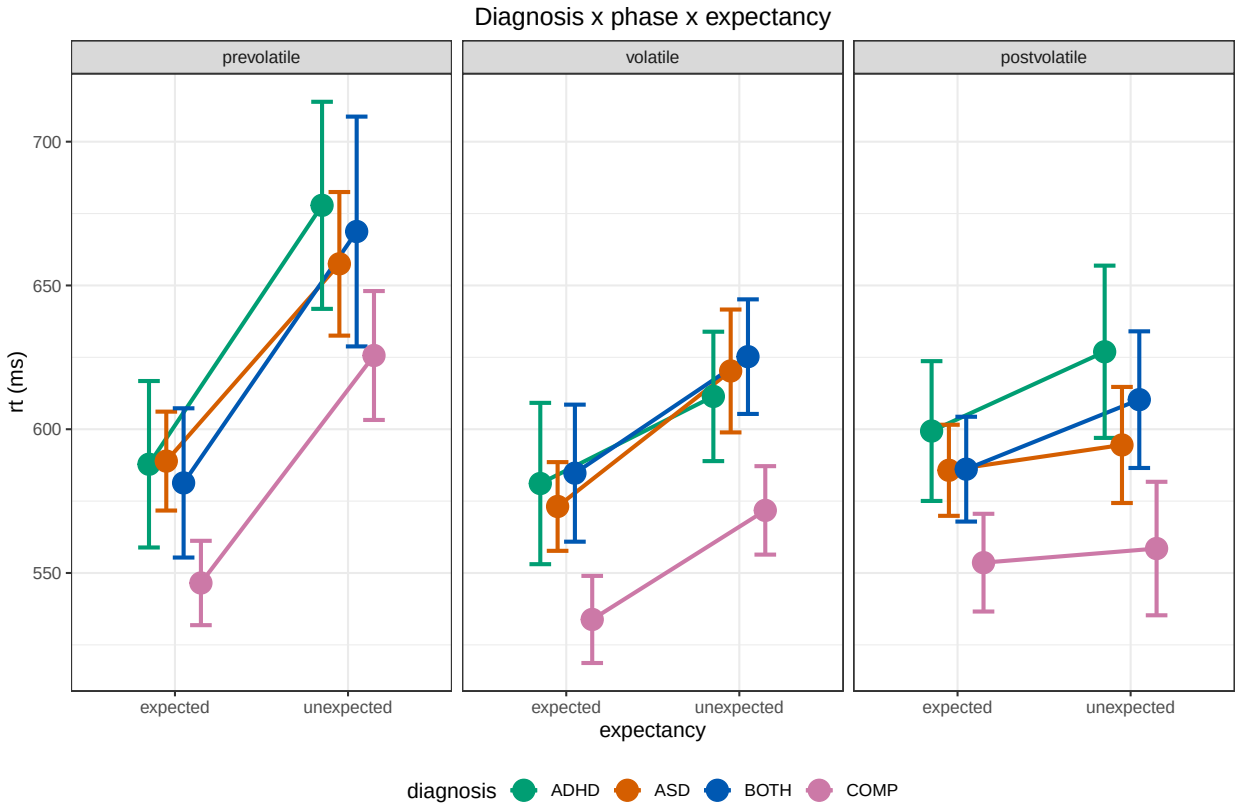



```
p5 = df.pal %>%
  group_by(subID, expected, phase) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  )  %>%
  group_by(phase, expected) %>%
```

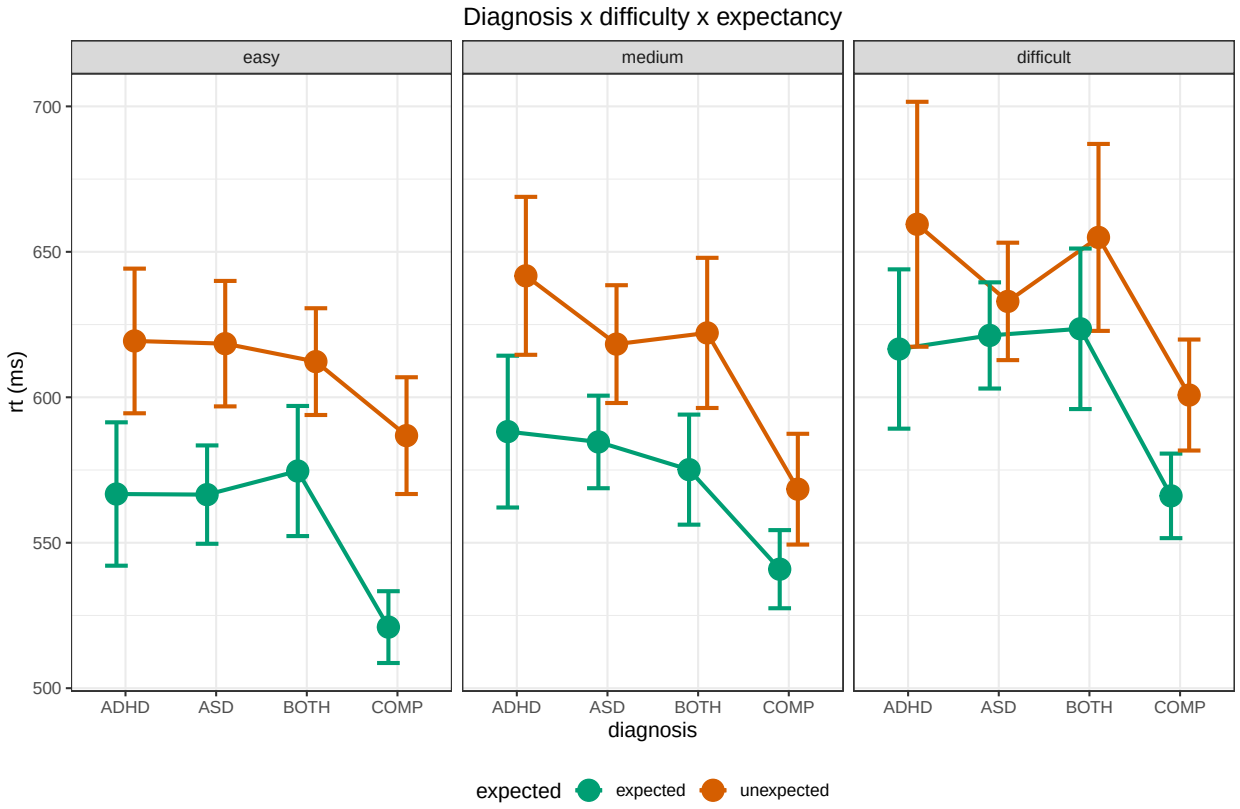
```
  summarise(
    rt.mn = mean(rt.cor),
    rt.se = sd(rt.cor) / sqrt(n())
  ) %>%
  ggplot(aes(y = rt.mn, x = phase, group = expected, colour = expected)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
  geom_errorbar(aes(ymin = rt.mn - rt.se,
                    ymax = rt.mn + rt.se), 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)
```



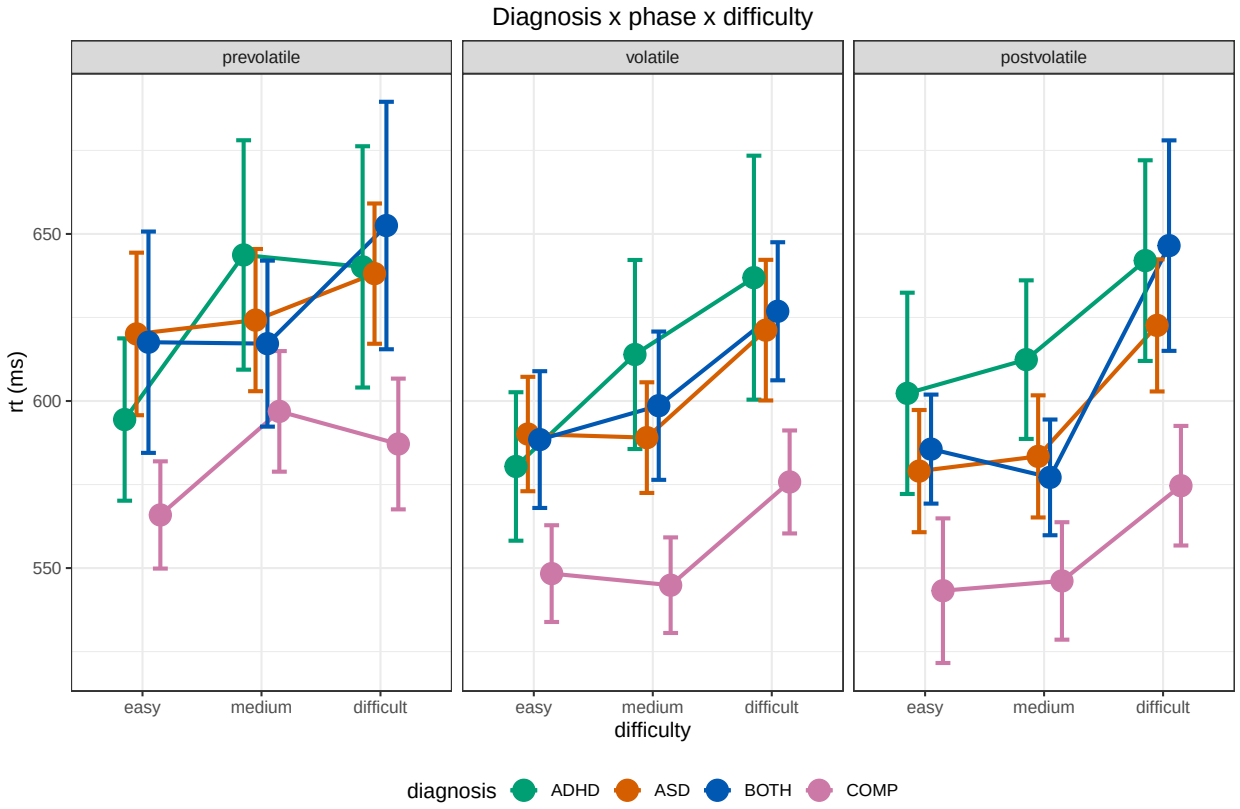

```r
# three-way interaction
df.pal %>%
  group_by(subID, diagnosis, expected, phase) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  )  %>%
  group_by(diagnosis, expected, phase) %>%
  summarise(
    rt.mn = mean(rt.cor),
    rt.se = sd(rt.cor) / sqrt(n())
  ) %>%
  ggplot(aes(y = rt.mn, x = 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))
```



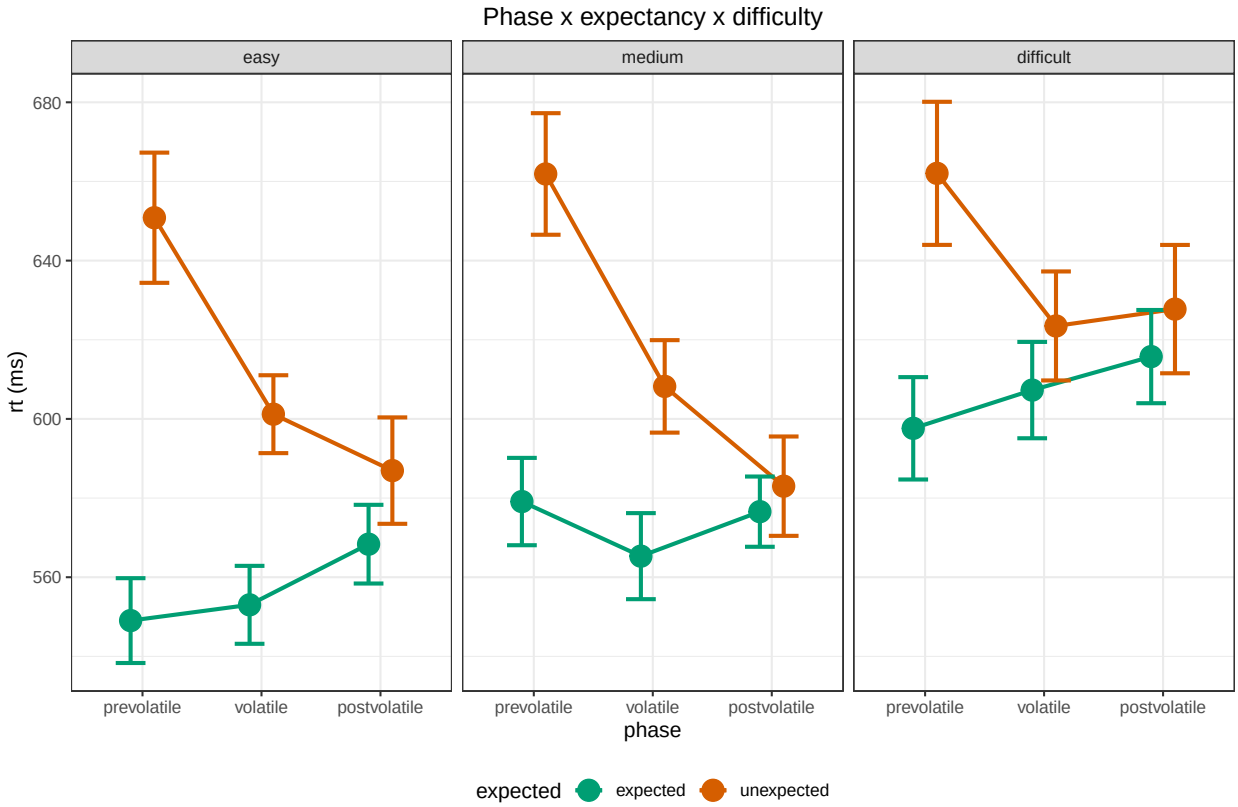

```
df.pal %>%
  group_by(subID, diagnosis, expected, difficulty) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  )  %>%
  group_by(diagnosis, expected, difficulty) %>%
  summarise(
    rt.mn = mean(rt.cor),
    rt.se = sd(rt.cor) / sqrt(n())
  ) %>%
  ggplot(aes(y = rt.mn, x = diagnosis, group = 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))
```

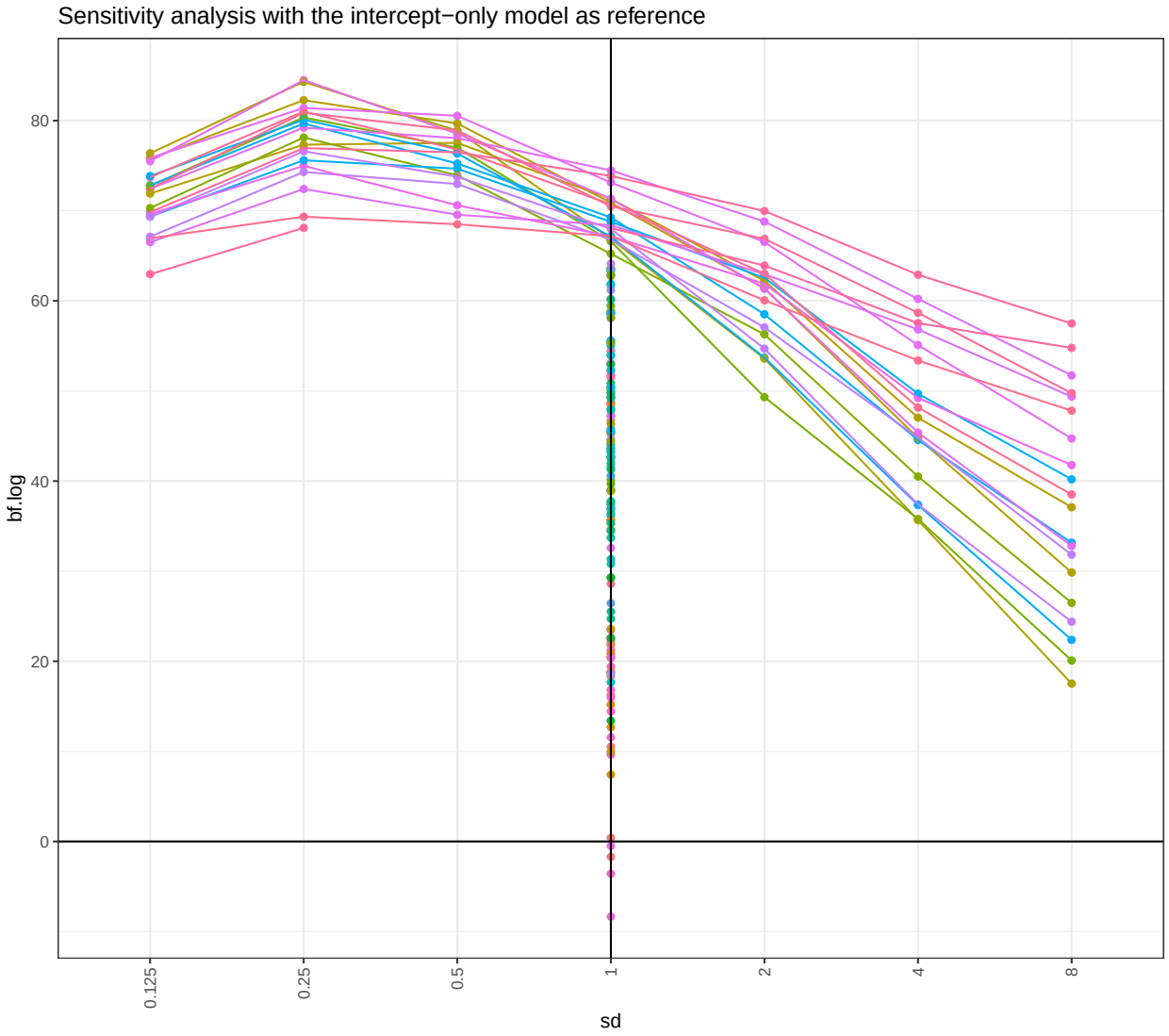


Bayes factor analysis

```
# 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
set.seed(5544)
m.pal.bf = brm(f.pal,
          df.pal, prior = priors,
          family = shifted_lognormal,
          iter = 40000, warmup = 10000,
          backend = "cmdstanr", threads = threading(8),
          file = file.path(brms_dir, "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
                reps = 5
    )
  }
  },
  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" & !is.na(bf.log)) %>%
 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(
  "sdx2",    "sdx4",   "sdx8",
  "sdx0.5", "sdx0.25", "sdx0.125"
  )

# 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))
  }
  )
}
```

```r
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))
```



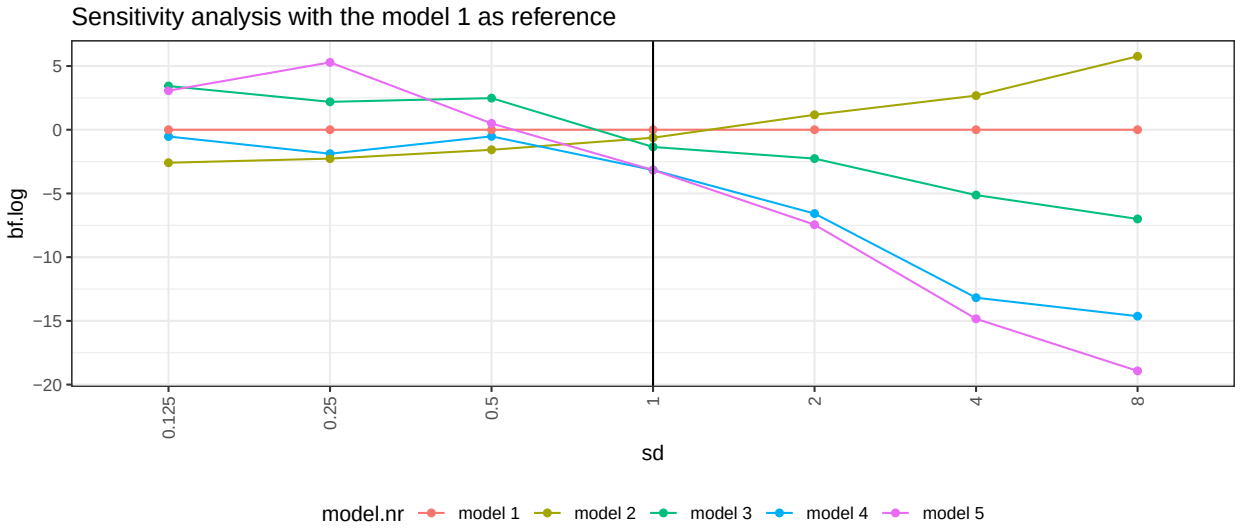
There seems to be a bit of a change between the best few models from the chosen prior to narrower or wider priors, let's inspect this more closely.

```
# chose the top 4 models for closer inspection
df.pal.bf.best = df.pal.bf %>%
  drop_na() %>%
  # 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 < 6) %>%
  # 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(-`population-level`) %>%
  pivot_wider(names_from = priors, values_from = bf.log) %>%
    arrange(desc(chosen)) %>%
    relocate(model.nr, `sdx0.125`, `sdx0.25`, `sdx0.5`, `chosen`, `sdx2`, `sdx4`, `sdx8`)
  )
```

model.nr	sdx0.125	sdx0.25	sdx0.5	chosen	sdx2	sdx4	sdx8
model 1	72.4367	79.2156	78.0568	74.4847	68.7962	60.2180	51.7267
model 2	69.8482	76.9495	76.4820	73.8563	69.9708	62.8958	57.4885
model 3	75.8675	81.4059	80.5378	73.1331	66.5321	55.0912	44.7257
model 4	71.9045	77.3350	77.5374	71.3240	62.2166	47.0348	37.0936
model 5	75.5027	84.5011	78.5547	71.3230	61.3441	45.3799	32.8017

```
# use the model which is best at chosen priors as comparison model
df.pal.bf.best  %>%
  group_by(priors) %>%
  mutate(bf.log = bf.log - bf.log[`model.nr` == "model 1"]) %>%
  ungroup() %>%
  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 model 1 as reference") +
  theme_bw() +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```




```
# check out which model is which
kable(df.pal.bf.best %>%
        select(model.nr, `population-level`) %>% distinct() %>%
        arrange(model.nr))
```

model.nr	population-level
model 1	diagnosis + expected + phase + difficulty + expected:phase + expected:difficulty
model 2	expected + phase + difficulty + expected:phase + expected:difficulty
model 3	diagnosis + expected + phase + difficulty + expected:phase + expected:difficulty + phase:difficulty
model 4	diagnosis + expected + phase + difficulty + diagnosis:difficulty + expected:phase + expected:difficulty
model 5	diagnosis + expected + phase + difficulty + expected:phase + expected:difficulty + phase:difficulty + expected:phase:difficulty

None of the best five models includes a relevant interaction with the diagnostic group. Therefore, this conventional analysis of reaction time does not suggest any differences in probabilistic learning.

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 = file.path(brms_dir, "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: 1620)
##   Draws: 4 chains, each with iter = 6000; warmup = 1500; thin = 1;
##          total post-warmup draws = 18000
##
## Multilevel Hyperparameters:
## ~subID (Number of levels: 90)
##                                                 Estimate Est.Error l-95% CI
## sd(Intercept)                                       0.24      0.02     0.20
```

```
## sd(expected1)                                  0.02      0.01      0.00
## sd(phase1)                                     0.08      0.01      0.07
## sd(phase2)                                     0.03      0.01      0.02
## sd(difficulty1)                                0.01      0.01      0.00
## sd(difficulty2)                                0.02      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)                         0.02      0.01      0.00
## sd(phase2:difficulty2)                         0.04      0.01      0.02
## sd(expected1:difficulty1)                      0.01      0.00      0.00
## sd(expected1:difficulty2)                      0.02      0.01      0.00
## cor(Intercept,expected1)                      -0.08      0.18     -0.41
## cor(Intercept,phase1)                          0.12      0.10     -0.09
## cor(expected1,phase1)                          0.19      0.18     -0.19
## cor(Intercept,phase2)                         -0.04      0.14     -0.32
## cor(expected1,phase2)                          0.10      0.21     -0.34
## cor(phase1,phase2)                            -0.28      0.15     -0.56
## cor(Intercept,difficulty1)                    -0.23      0.21     -0.60
## cor(expected1,difficulty1)                    -0.01      0.23     -0.46
## cor(phase1,difficulty1)                       -0.05      0.21     -0.45
## cor(phase2,difficulty1)                       -0.15      0.23     -0.55
## cor(Intercept,difficulty2)                    -0.17      0.20     -0.52
## cor(expected1,difficulty2)                    -0.12      0.23     -0.54
## cor(phase1,difficulty2)                       -0.25      0.20     -0.60
## cor(phase2,difficulty2)                       -0.01      0.22     -0.43
## cor(difficulty1,difficulty2)                   0.00      0.24     -0.44
## cor(Intercept,expected1:phase1)               -0.07      0.17     -0.39
## cor(expected1,expected1:phase1)                0.03      0.22     -0.40
## cor(phase1,expected1:phase1)                  -0.32      0.17     -0.62
## cor(phase2,expected1:phase1)                  -0.03      0.21     -0.43
## cor(difficulty1,expected1:phase1)              0.01      0.23     -0.43
## cor(difficulty2,expected1:phase1)              0.07      0.22     -0.36
## cor(Intercept,expected1:phase2)                0.24      0.18     -0.15
## cor(expected1,expected1:phase2)               -0.09      0.22     -0.50
## cor(phase1,expected1:phase2)                   0.02      0.19     -0.35
## cor(phase2,expected1:phase2)                  -0.36      0.21     -0.70
## cor(difficulty1,expected1:phase2)              0.02      0.23     -0.42
## cor(difficulty2,expected1:phase2)              0.03      0.22     -0.41
## cor(expected1:phase1,expected1:phase2)        -0.02      0.22     -0.43
## cor(Intercept,phase1:difficulty1)             -0.25      0.14     -0.52
## cor(expected1,phase1:difficulty1)              0.03      0.21     -0.37
## cor(phase1,phase1:difficulty1)                 0.14      0.15     -0.17
## cor(phase2,phase1:difficulty1)                -0.03      0.19     -0.41
## cor(difficulty1,phase1:difficulty1)            0.09      0.23     -0.38
## cor(difficulty2,phase1:difficulty1)            0.03      0.22     -0.39
## cor(expected1:phase1,phase1:difficulty1)      -0.14      0.20     -0.52
## cor(expected1:phase2,phase1:difficulty1)      -0.08      0.21     -0.48
## cor(Intercept,phase2:difficulty1)             -0.16      0.20     -0.52
## cor(expected1,phase2:difficulty1)             -0.05      0.23     -0.49
## cor(phase1,phase2:difficulty1)                -0.08      0.20     -0.46
## cor(phase2,phase2:difficulty1)                -0.10      0.22     -0.52
## cor(difficulty1,phase2:difficulty1)            0.06      0.24     -0.41
```

```
## cor(difficulty2,phase2:difficulty1)                  0.12      0.23    -0.35
## cor(expected1:phase1,phase2:difficulty1)             0.08      0.23    -0.37
## cor(expected1:phase2,phase2:difficulty1)             0.11      0.23    -0.36
## cor(phase1:difficulty1,phase2:difficulty1)          -0.02      0.22    -0.44
## cor(Intercept,phase1:difficulty2)                    0.05      0.20    -0.35
## cor(expected1,phase1:difficulty2)                    0.08      0.23    -0.38
## cor(phase1,phase1:difficulty2)                       0.01      0.20    -0.39
## cor(phase2,phase1:difficulty2)                       0.14      0.23    -0.33
## cor(difficulty1,phase1:difficulty2)                 -0.09      0.24    -0.54
## cor(difficulty2,phase1:difficulty2)                 -0.11      0.23    -0.54
## cor(expected1:phase1,phase1:difficulty2)            -0.06      0.23    -0.49
## cor(expected1:phase2,phase1:difficulty2)            -0.09      0.23    -0.52
## cor(phase1:difficulty1,phase1:difficulty2)          -0.10      0.23    -0.52
## cor(phase2:difficulty1,phase1:difficulty2)          -0.14      0.24    -0.57
## cor(Intercept,phase2:difficulty2)                    0.27      0.14    -0.01
## cor(expected1,phase2:difficulty2)                    0.10      0.21    -0.33
## cor(phase1,phase2:difficulty2)                      -0.04      0.15    -0.34
## cor(phase2,phase2:difficulty2)                       0.05      0.19    -0.33
## cor(difficulty1,phase2:difficulty2)                 -0.12      0.23    -0.53
## cor(difficulty2,phase2:difficulty2)                 -0.23      0.22    -0.61
## cor(expected1:phase1,phase2:difficulty2)             0.08      0.20    -0.32
## cor(expected1:phase2,phase2:difficulty2)             0.12      0.21    -0.30
## cor(phase1:difficulty1,phase2:difficulty2)          -0.17      0.19    -0.52
## cor(phase2:difficulty1,phase2:difficulty2)          -0.16      0.23    -0.57
## cor(phase1:difficulty2,phase2:difficulty2)           0.08      0.22    -0.36
## cor(Intercept,expected1:difficulty1)                 0.09      0.24    -0.39
## cor(expected1,expected1:difficulty1)                 0.02      0.24    -0.45
## cor(phase1,expected1:difficulty1)                    0.04      0.23    -0.42
## cor(phase2,expected1:difficulty1)                    0.01      0.24    -0.45
## cor(difficulty1,expected1:difficulty1)              -0.07      0.25    -0.53
## cor(difficulty2,expected1:difficulty1)              -0.02      0.24    -0.48
## cor(expected1:phase1,expected1:difficulty1)          0.02      0.24    -0.44
## 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)        0.02      0.24    -0.46
## cor(phase1:difficulty2,expected1:difficulty1)       -0.01      0.24    -0.47
## cor(phase2:difficulty2,expected1:difficulty1)        0.01      0.24    -0.44
## cor(Intercept,expected1:difficulty2)                -0.02      0.18    -0.36
## cor(expected1,expected1:difficulty2)                 0.16      0.22    -0.30
## cor(phase1,expected1:difficulty2)                    0.36      0.19    -0.06
## cor(phase2,expected1:difficulty2)                   -0.07      0.21    -0.47
## cor(difficulty1,expected1:difficulty2)              -0.01      0.23    -0.46
## cor(difficulty2,expected1:difficulty2)              -0.29      0.24    -0.68
## cor(expected1:phase1,expected1:difficulty2)         -0.04      0.22    -0.45
## cor(expected1:phase2,expected1:difficulty2)         -0.02      0.22    -0.44
## cor(phase1:difficulty1,expected1:difficulty2)       -0.07      0.21    -0.47
## cor(phase2:difficulty1,expected1:difficulty2)       -0.08      0.23    -0.51
## cor(phase1:difficulty2,expected1:difficulty2)        0.06      0.23    -0.39
## cor(phase2:difficulty2,expected1:difficulty2)        0.13      0.21    -0.30
## cor(expected1:difficulty1,expected1:difficulty2)    -0.08      0.25    -0.54
##                                                  u-95% CI Rhat Bulk_ESS
## sd(Intercept)                                        0.28 1.00     2016
## sd(expected1)                                        0.03 1.00     2555
## sd(phase1)                                           0.10 1.00     5670
```

```
## sd(phase2)                                          0.05 1.00     3112
## sd(difficulty1)                                     0.03 1.00     4286
## sd(difficulty2)                                     0.03 1.00     3195
## sd(expected1:phase1)                                0.04 1.00     4753
## sd(expected1:phase2)                                0.03 1.00     2762
## sd(phase1:difficulty1)                              0.06 1.00     6502
## sd(phase2:difficulty1)                              0.04 1.00     2875
## sd(phase1:difficulty2)                              0.04 1.00     2974
## sd(phase2:difficulty2)                              0.06 1.00     5613
## sd(expected1:difficulty1)                           0.02 1.00     7151
## sd(expected1:difficulty2)                           0.03 1.00     4804
## cor(Intercept,expected1)                            0.28 1.00    17272
## cor(Intercept,phase1)                               0.31 1.00     5750
## cor(expected1,phase1)                               0.53 1.00     1222
## cor(Intercept,phase2)                               0.25 1.00    13694
## cor(expected1,phase2)                               0.50 1.00     4262
## cor(phase1,phase2)                                  0.05 1.00    11038
## cor(Intercept,difficulty1)                          0.24 1.00    12961
## cor(expected1,difficulty1)                          0.44 1.00    14038
## cor(phase1,difficulty1)                             0.37 1.00    18307
## cor(phase2,difficulty1)                             0.32 1.00    13367
## cor(Intercept,difficulty2)                          0.25 1.00    14732
## cor(expected1,difficulty2)                          0.34 1.00     7567
## cor(phase1,difficulty2)                             0.18 1.00    13752
## cor(phase2,difficulty2)                             0.42 1.00    13536
## cor(difficulty1,difficulty2)                        0.47 1.00     9510
## cor(Intercept,expected1:phase1)                     0.26 1.00    15663
## cor(expected1,expected1:phase1)                     0.45 1.00     5095
## cor(phase1,expected1:phase1)                        0.04 1.00    14282
## cor(phase2,expected1:phase1)                        0.37 1.00    11451
## cor(difficulty1,expected1:phase1)                   0.46 1.00     8748
## cor(difficulty2,expected1:phase1)                   0.49 1.00     9388
## cor(Intercept,expected1:phase2)                     0.57 1.00    13776
## cor(expected1,expected1:phase2)                     0.35 1.00     7315
## cor(phase1,expected1:phase2)                        0.39 1.00    15505
## cor(phase2,expected1:phase2)                        0.12 1.00     5176
## cor(difficulty1,expected1:phase2)                   0.46 1.00    12450
## cor(difficulty2,expected1:phase2)                   0.45 1.00    12500
## cor(expected1:phase1,expected1:phase2)              0.42 1.00    13042
## cor(Intercept,phase1:difficulty1)                   0.03 1.00    15675
## cor(expected1,phase1:difficulty1)                   0.44 1.00     5204
## cor(phase1,phase1:difficulty1)                      0.43 1.00    13980
## cor(phase2,phase1:difficulty1)                      0.35 1.00     7226
## cor(difficulty1,phase1:difficulty1)                 0.51 1.00     5101
## cor(difficulty2,phase1:difficulty1)                 0.45 1.00     6500
## cor(expected1:phase1,phase1:difficulty1)            0.28 1.00     6737
## cor(expected1:phase2,phase1:difficulty1)            0.33 1.00     7166
## cor(Intercept,phase2:difficulty1)                   0.27 1.00    15437
## cor(expected1,phase2:difficulty1)                   0.41 1.00    11545
## cor(phase1,phase2:difficulty1)                      0.33 1.00    17374
## cor(phase2,phase2:difficulty1)                      0.35 1.00    11916
## cor(difficulty1,phase2:difficulty1)                 0.50 1.00    10629
## cor(difficulty2,phase2:difficulty1)                 0.55 1.00     9787
## cor(expected1:phase1,phase2:difficulty1)            0.50 1.00    12243
```

```
## cor(expected1:phase2,phase2:difficulty1)             0.53 1.00     9544
## cor(phase1:difficulty1,phase2:difficulty1)           0.43 1.00    13701
## cor(Intercept,phase1:difficulty2)                    0.43 1.00    19093
## cor(expected1,phase1:difficulty2)                    0.52 1.00    10317
## cor(phase1,phase1:difficulty2)                       0.41 1.00    19241
## cor(phase2,phase1:difficulty2)                       0.55 1.00     9999
## cor(difficulty1,phase1:difficulty2)                  0.39 1.00     8913
## cor(difficulty2,phase1:difficulty2)                  0.36 1.00    10078
## cor(expected1:phase1,phase1:difficulty2)             0.39 1.00    13384
## cor(expected1:phase2,phase1:difficulty2)             0.37 1.00    10128
## cor(phase1:difficulty1,phase1:difficulty2)           0.37 1.00    12352
## cor(phase2:difficulty1,phase1:difficulty2)           0.35 1.00     8561
## cor(Intercept,phase2:difficulty2)                    0.53 1.00    16648
## cor(expected1,phase2:difficulty2)                    0.49 1.00     5397
## cor(phase1,phase2:difficulty2)                       0.26 1.00    14644
## cor(phase2,phase2:difficulty2)                       0.41 1.00     7624
## cor(difficulty1,phase2:difficulty2)                  0.34 1.00     5051
## cor(difficulty2,phase2:difficulty2)                  0.23 1.00     4437
## cor(expected1:phase1,phase2:difficulty2)             0.46 1.00     8301
## cor(expected1:phase2,phase2:difficulty2)             0.52 1.00     7230
## cor(phase1:difficulty1,phase2:difficulty2)           0.21 1.00    10482
## cor(phase2:difficulty1,phase2:difficulty2)           0.31 1.00     6875
## cor(phase1:difficulty2,phase2:difficulty2)           0.51 1.00     7432
## cor(Intercept,expected1:difficulty1)                 0.53 1.00    22663
## cor(expected1,expected1:difficulty1)                 0.48 1.00    21005
## cor(phase1,expected1:difficulty1)                    0.49 1.00    24596
## cor(phase2,expected1:difficulty1)                    0.47 1.00    20325
## cor(difficulty1,expected1:difficulty1)               0.42 1.00    14764
## cor(difficulty2,expected1:difficulty1)               0.45 1.00    17245
## cor(expected1:phase1,expected1:difficulty1)          0.48 1.00    18140
## cor(expected1:phase2,expected1:difficulty1)          0.49 1.00    16000
## cor(phase1:difficulty1,expected1:difficulty1)        0.49 1.00    17353
## cor(phase2:difficulty1,expected1:difficulty1)        0.48 1.00    13519
## cor(phase1:difficulty2,expected1:difficulty1)        0.45 1.00    14865
## cor(phase2:difficulty2,expected1:difficulty1)        0.47 1.00    18171
## cor(Intercept,expected1:difficulty2)                 0.34 1.00    19887
## cor(expected1,expected1:difficulty2)                 0.56 1.00     5382
## cor(phase1,expected1:difficulty2)                    0.67 1.00    12163
## cor(phase2,expected1:difficulty2)                    0.35 1.00    13215
## cor(difficulty1,expected1:difficulty2)               0.45 1.00     9560
## cor(difficulty2,expected1:difficulty2)               0.23 1.00     5067
## cor(expected1:phase1,expected1:difficulty2)          0.39 1.00    13631
## cor(expected1:phase2,expected1:difficulty2)          0.41 1.00    14253
## cor(phase1:difficulty1,expected1:difficulty2)        0.35 1.00    16180
## cor(phase2:difficulty1,expected1:difficulty2)        0.38 1.00    11489
## cor(phase1:difficulty2,expected1:difficulty2)        0.49 1.00    13311
## cor(phase2:difficulty2,expected1:difficulty2)        0.52 1.00    14802
## cor(expected1:difficulty1,expected1:difficulty2)     0.42 1.00    11514
##                                                  Tail_ESS
## sd(Intercept)                                        4222
## sd(expected1)                                        2666
## sd(phase1)                                           9882
## sd(phase2)                                           2140
## sd(difficulty1)                                      5586
```

```
## sd(difficulty2)                                    3044
## sd(expected1:phase1)                               3922
## sd(expected1:phase2)                               2058
## sd(phase1:difficulty1)                             7953
## sd(phase2:difficulty1)                             4258
## sd(phase1:difficulty2)                             4460
## sd(phase2:difficulty2)                             6427
## sd(expected1:difficulty1)                          7363
## sd(expected1:difficulty2)                          3837
## cor(Intercept,expected1)                          12160
## cor(Intercept,phase1)                              9520
## cor(expected1,phase1)                              2084
## cor(Intercept,phase2)                             13477
## cor(expected1,phase2)                              6958
## cor(phase1,phase2)                                 8230
## cor(Intercept,difficulty1)                        10868
## cor(expected1,difficulty1)                        14545
## cor(phase1,difficulty1)                           12468
## cor(phase2,difficulty1)                           13434
## cor(Intercept,difficulty2)                        11012
## cor(expected1,difficulty2)                        10456
## cor(phase1,difficulty2)                           10669
## cor(phase2,difficulty2)                           13665
## cor(difficulty1,difficulty2)                      13007
## cor(Intercept,expected1:phase1)                   13293
## cor(expected1,expected1:phase1)                   10212
## cor(phase1,expected1:phase1)                      10980
## cor(phase2,expected1:phase1)                      13421
## cor(difficulty1,expected1:phase1)                 12299
## cor(difficulty2,expected1:phase1)                 12917
## cor(Intercept,expected1:phase2)                    9394
## cor(expected1,expected1:phase2)                   10453
## cor(phase1,expected1:phase2)                      13307
## cor(phase2,expected1:phase2)                       5389
## cor(difficulty1,expected1:phase2)                 12928
## cor(difficulty2,expected1:phase2)                 14410
## cor(expected1:phase1,expected1:phase2)            13832
## cor(Intercept,phase1:difficulty1)                 13050
## cor(expected1,phase1:difficulty1)                  9530
## cor(phase1,phase1:difficulty1)                    14051
## cor(phase2,phase1:difficulty1)                    11668
## cor(difficulty1,phase1:difficulty1)                7570
## cor(difficulty2,phase1:difficulty1)               10226
## cor(expected1:phase1,phase1:difficulty1)           9211
## cor(expected1:phase2,phase1:difficulty1)          11176
## cor(Intercept,phase2:difficulty1)                 11267
## cor(expected1,phase2:difficulty1)                 12744
## cor(phase1,phase2:difficulty1)                    13335
## cor(phase2,phase2:difficulty1)                    12758
## cor(difficulty1,phase2:difficulty1)               13035
## cor(difficulty2,phase2:difficulty1)               11821
## cor(expected1:phase1,phase2:difficulty1)          13885
## cor(expected1:phase2,phase2:difficulty1)          12099
## cor(phase1:difficulty1,phase2:difficulty1)        14714
```

```
## cor(Intercept,phase1:difficulty2)                    11843
## cor(expected1,phase1:difficulty2)                    12128
## cor(phase1,phase1:difficulty2)                       13603
## cor(phase2,phase1:difficulty2)                       12163
## cor(difficulty1,phase1:difficulty2)                  12675
## cor(difficulty2,phase1:difficulty2)                  12614
## cor(expected1:phase1,phase1:difficulty2)             14015
## cor(expected1:phase2,phase1:difficulty2)             13232
## cor(phase1:difficulty1,phase1:difficulty2)           12446
## cor(phase2:difficulty1,phase1:difficulty2)           13105
## cor(Intercept,phase2:difficulty2)                    13778
## cor(expected1,phase2:difficulty2)                     8884
## cor(phase1,phase2:difficulty2)                       13339
## cor(phase2,phase2:difficulty2)                       11574
## cor(difficulty1,phase2:difficulty2)                   9045
## cor(difficulty2,phase2:difficulty2)                   7077
## cor(expected1:phase1,phase2:difficulty2)             12240
## cor(expected1:phase2,phase2:difficulty2)              9523
## cor(phase1:difficulty1,phase2:difficulty2)           13564
## cor(phase2:difficulty1,phase2:difficulty2)           11080
## cor(phase1:difficulty2,phase2:difficulty2)           11170
## cor(Intercept,expected1:difficulty1)                 12037
## cor(expected1,expected1:difficulty1)                 13611
## cor(phase1,expected1:difficulty1)                    13342
## cor(phase2,expected1:difficulty1)                    13832
## cor(difficulty1,expected1:difficulty1)               12979
## cor(difficulty2,expected1:difficulty1)               14167
## cor(expected1:phase1,expected1:difficulty1)          13617
## cor(expected1:phase2,expected1:difficulty1)          14690
## cor(phase1:difficulty1,expected1:difficulty1)        14183
## cor(phase2:difficulty1,expected1:difficulty1)        13892
## cor(phase1:difficulty2,expected1:difficulty1)        14792
## cor(phase2:difficulty2,expected1:difficulty1)        14580
## cor(Intercept,expected1:difficulty2)                 11951
## cor(expected1,expected1:difficulty2)                  7595
## cor(phase1,expected1:difficulty2)                     8330
## cor(phase2,expected1:difficulty2)                    12885
## cor(difficulty1,expected1:difficulty2)               13211
## cor(difficulty2,expected1:difficulty2)                8547
## cor(expected1:phase1,expected1:difficulty2)          14135
## cor(expected1:phase2,expected1:difficulty2)          14484
## cor(phase1:difficulty1,expected1:difficulty2)        14284
## cor(phase2:difficulty1,expected1:difficulty2)        14826
## cor(phase1:difficulty2,expected1:difficulty2)        13822
## cor(phase2:difficulty2,expected1:difficulty2)        14984
## cor(expected1:difficulty1,expected1:difficulty2)     14329
##
## Regression Coefficients:
##                                      Estimate Est.Error l-95% CI u-95% CI
## Intercept                                5.91      0.04     5.83     6.00
## diagnosis1                               0.02      0.03    -0.04     0.07
## diagnosis2                               0.03      0.03    -0.02     0.09
## diagnosis3                               0.01      0.03    -0.04     0.07
## expected1                               -0.04      0.00    -0.05    -0.04
```

```
## phase1                                0.03      0.01     0.01     0.05
## phase2                               -0.01      0.01    -0.02    -0.00
## difficulty1                          -0.04      0.00    -0.05    -0.03
## difficulty2                          -0.01      0.00    -0.02    -0.00
## diagnosis1:expected1                 -0.00      0.01    -0.01     0.01
## diagnosis2:expected1                  0.00      0.01    -0.01     0.02
## diagnosis3:expected1                 -0.00      0.01    -0.01     0.01
## diagnosis1:phase1                    -0.03      0.02    -0.06     0.00
## diagnosis2:phase1                     0.01      0.02    -0.02     0.04
## diagnosis3:phase1                    -0.00      0.01    -0.03     0.03
## diagnosis1:phase2                    -0.00      0.01    -0.02     0.02
## diagnosis2:phase2                     0.00      0.01    -0.02     0.02
## diagnosis3:phase2                    -0.00      0.01    -0.02     0.02
## expected1:phase1                     -0.04      0.01    -0.05    -0.03
## expected1:phase2                      0.00      0.00    -0.01     0.01
## diagnosis1:difficulty1               -0.02      0.01    -0.03     0.00
## diagnosis2:difficulty1                0.01      0.01    -0.01     0.02
## diagnosis3:difficulty1                0.01      0.01    -0.00     0.03
## diagnosis1:difficulty2                0.02      0.01     0.01     0.04
## diagnosis2:difficulty2               -0.01      0.01    -0.02     0.01
## diagnosis3:difficulty2               -0.03      0.01    -0.04    -0.01
## expected1:difficulty1                -0.02      0.00    -0.03    -0.01
## expected1:difficulty2                 0.00      0.00    -0.01     0.01
## phase1:difficulty1                   -0.01      0.01    -0.02     0.01
## phase2:difficulty1                    0.00      0.01    -0.01     0.02
## phase1:difficulty2                    0.02      0.01     0.01     0.04
## phase2:difficulty2                   -0.00      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     0.03
## diagnosis3:expected1:phase1          -0.00      0.01    -0.02     0.01
## diagnosis1:expected1:phase2          -0.00      0.01    -0.02     0.01
## diagnosis2:expected1:phase2          -0.01      0.01    -0.02     0.01
## diagnosis3:expected1:phase2           0.01      0.01    -0.01     0.03
## diagnosis1:expected1:difficulty1     -0.00      0.01    -0.02     0.01
## diagnosis2:expected1:difficulty1     -0.01      0.01    -0.02     0.01
## diagnosis3:expected1:difficulty1      0.01      0.01    -0.01     0.02
## diagnosis1:expected1:difficulty2     -0.00      0.01    -0.02     0.01
## diagnosis2:expected1:difficulty2      0.00      0.01    -0.01     0.02
## diagnosis3:expected1:difficulty2     -0.00      0.01    -0.02     0.01
## diagnosis1:phase1:difficulty1        -0.02      0.01    -0.04     0.01
## diagnosis2:phase1:difficulty1         0.01      0.01    -0.01     0.04
## diagnosis3:phase1:difficulty1         0.01      0.01    -0.02     0.03
## diagnosis1:phase2:difficulty1        -0.00      0.01    -0.02     0.02
## diagnosis2:phase2:difficulty1         0.00      0.01    -0.02     0.02
## diagnosis3:phase2:difficulty1        -0.01      0.01    -0.03     0.01
## diagnosis1:phase1:difficulty2        -0.00      0.01    -0.02     0.02
## diagnosis2:phase1:difficulty2        -0.01      0.01    -0.03     0.01
## diagnosis3:phase1:difficulty2        -0.00      0.01    -0.03     0.02
## diagnosis1:phase2:difficulty2         0.00      0.01    -0.02     0.03
## diagnosis2:phase2:difficulty2         0.01      0.01    -0.02     0.03
## diagnosis3:phase2:difficulty2         0.01      0.01    -0.01     0.04
## 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     0.01
```

```
## expected1:phase2:difficulty2                -0.01      0.01    -0.02     0.00
## diagnosis1:expected1:phase1:difficulty1      0.00      0.01    -0.02     0.02
## diagnosis2:expected1:phase1:difficulty1      0.01      0.01    -0.01     0.03
## diagnosis3:expected1:phase1:difficulty1      0.00      0.01    -0.02     0.02
## diagnosis1:expected1:phase2:difficulty1      0.00      0.01    -0.02     0.02
## diagnosis2:expected1:phase2:difficulty1      0.00      0.01    -0.02     0.02
## diagnosis3:expected1:phase2:difficulty1     -0.01      0.01    -0.03     0.01
## diagnosis1:expected1:phase1:difficulty2      0.01      0.01    -0.01     0.03
## diagnosis2:expected1:phase1:difficulty2     -0.00      0.01    -0.02     0.02
## diagnosis3: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     -0.01      0.01    -0.03     0.01
## diagnosis3:expected1:phase2:difficulty2      0.01      0.01    -0.01     0.03
##                                           Rhat Bulk_ESS Tail_ESS
## Intercept                                 1.00     1730     4745
## diagnosis1                                1.00     1920     4082
## diagnosis2                                1.00     1821     3695
## diagnosis3                                1.00     1813     3714
## expected1                                 1.00    16117    13497
## phase1                                    1.00     5220     8780
## phase2                                    1.00    12983    13534
## difficulty1                               1.00    15260    13656
## difficulty2                               1.00    19463    14630
## diagnosis1:expected1                      1.00    15109    12506
## diagnosis2:expected1                      1.00    15619    13499
## diagnosis3:expected1                      1.00    16213    14154
## diagnosis1:phase1                         1.00     5636     8934
## diagnosis2:phase1                         1.00     5510     8725
## diagnosis3:phase1                         1.00     5725     9627
## diagnosis1:phase2                         1.00    11890    13046
## diagnosis2:phase2                         1.00    10937    13345
## diagnosis3:phase2                         1.00    11717    12959
## expected1:phase1                          1.00    14200    14074
## expected1:phase2                          1.00    18159    14291
## diagnosis1:difficulty1                    1.00    15332    14442
## diagnosis2:difficulty1                    1.00    15715    13896
## diagnosis3:difficulty1                    1.00    15840    13587
## diagnosis1:difficulty2                    1.00    13952    13979
## diagnosis2:difficulty2                    1.00    14665    13901
## diagnosis3:difficulty2                    1.00    13716    13655
## expected1:difficulty1                     1.00    21076    15241
## expected1:difficulty2                     1.00    18862    14430
## phase1:difficulty1                        1.00    11191    12745
## phase2:difficulty1                        1.00    16365    14555
## phase1:difficulty2                        1.00    15415    14127
## phase2:difficulty2                        1.00    11999    13288
## diagnosis1:expected1:phase1               1.00    14174    13639
## diagnosis2:expected1:phase1               1.00    13919    13850
## diagnosis3:expected1:phase1               1.00    13127    12941
## diagnosis1:expected1:phase2               1.00    14496    14407
## diagnosis2:expected1:phase2               1.00    14489    14469
## diagnosis3:expected1:phase2               1.00    13801    13893
## diagnosis1:expected1:difficulty1          1.00    17011    14346
## diagnosis2:expected1:difficulty1          1.00    15362    13792
```

```
## diagnosis3:expected1:difficulty1        1.00    16473    13863
## diagnosis1:expected1:difficulty2        1.00    13878    13361
## diagnosis2:expected1:difficulty2        1.00    13896    13978
## diagnosis3:expected1:difficulty2        1.00    14102    14265
## diagnosis1:phase1:difficulty1           1.00    11130    11657
## diagnosis2:phase1:difficulty1           1.00    10195    11649
## diagnosis3:phase1:difficulty1           1.00    10735    12917
## diagnosis1:phase2:difficulty1           1.00    13174    13788
## diagnosis2:phase2:difficulty1           1.00    13440    13567
## diagnosis3:phase2:difficulty1           1.00    13061    14118
## diagnosis1:phase1:difficulty2           1.00    14038    13762
## diagnosis2:phase1:difficulty2           1.00    12307    12692
## diagnosis3:phase1:difficulty2           1.00    13023    14373
## diagnosis1:phase2:difficulty2           1.00    11708    13036
## diagnosis2:phase2:difficulty2           1.00    11090    13154
## diagnosis3:phase2:difficulty2           1.00    11038    12251
## expected1:phase1:difficulty1            1.00    15967    13909
## expected1:phase2:difficulty1            1.00    16188    14010
## expected1:phase1:difficulty2            1.00    15335    14242
## expected1:phase2:difficulty2            1.00    15815    14317
## diagnosis1:expected1:phase1:difficulty1 1.00    12647    13041
## diagnosis2:expected1:phase1:difficulty1 1.00    12619    13425
## diagnosis3:expected1:phase1:difficulty1 1.00    12977    12232
## diagnosis1:expected1:phase2:difficulty1 1.00    12694    12849
## diagnosis2:expected1:phase2:difficulty1 1.00    13263    14366
## diagnosis3:expected1:phase2:difficulty1 1.00    12170    12995
## diagnosis1:expected1:phase1:difficulty2 1.00    12461    13777
## diagnosis2:expected1:phase1:difficulty2 1.00    12457    13301
## diagnosis3:expected1:phase1:difficulty2 1.00    13287    13572
## diagnosis1:expected1:phase2:difficulty2 1.00    12591    13358
## diagnosis2:expected1:phase2:difficulty2 1.00    12600    13937
## diagnosis3:expected1:phase2:difficulty2 1.00    12853    13725
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.13      0.01     0.12     0.14 1.00     4128     7712
## ndt     212.10     11.83   187.08   233.35 1.00     5825     8479
##
## 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	0.0160305	0.0208257	0.0047952
diagnosis1:difficulty1	-0.0152346	-0.0202866	-0.0050519
diagnosis1:difficulty2	0.0232305	0.0239342	0.0007036
diagnosis1:expected1	-0.0020182	-0.0080076	-0.0059895
diagnosis1:expected1:difficulty1	-0.0045934	0.0011407	0.0057341
diagnosis1:expected1:difficulty2	-0.0020800	-0.0019117	0.0001683
diagnosis1:expected1:phase1	0.0031739	-0.0000719	-0.0032459
diagnosis1:expected1:phase1:difficulty1	0.0037546	-0.0051538	-0.0089084
diagnosis1:expected1:phase1:difficulty2	0.0074234	0.0060690	-0.0013544
diagnosis1:expected1:phase2	-0.0045406	0.0018260	0.0063666
diagnosis1:expected1:phase2:difficulty1	0.0034878	0.0061304	0.0026426
diagnosis1:expected1:phase2:difficulty2	-0.0125092	-0.0123857	0.0001234
diagnosis1:phase1	-0.0271165	-0.0199222	0.0071943
diagnosis1:phase1:difficulty1	-0.0164401	-0.0064664	0.0099737
diagnosis1:phase1:difficulty2	-0.0003568	0.0005862	0.0009431
diagnosis1:phase2	-0.0016191	-0.0030865	-0.0014674
diagnosis1:phase2:difficulty1	-0.0005597	-0.0049369	-0.0043772
diagnosis1:phase2:difficulty2	0.0035350	-0.0000382	-0.0035732
diagnosis2	0.0316581	0.0309340	-0.0007241
diagnosis2:difficulty1	0.0066220	0.0109889	0.0043669
diagnosis2:difficulty2	-0.0057267	-0.0104102	-0.0046835
diagnosis2:expected1	0.0040875	0.0094331	0.0053456
diagnosis2:expected1:difficulty1	-0.0056077	-0.0103183	-0.0047105
diagnosis2:expected1:difficulty2	0.0044707	0.0087610	0.0042903
diagnosis2:expected1:phase1	0.0097832	0.0076629	-0.0021203
diagnosis2:expected1:phase1:difficulty1	0.0072642	0.0125615	0.0052973
diagnosis2:expected1:phase1:difficulty2	-0.0018248	-0.0052012	-0.0033764
diagnosis2:expected1:phase2	-0.0051722	-0.0053017	-0.0001296
diagnosis2:expected1:phase2:difficulty1	0.0045965	-0.0001899	-0.0047864
diagnosis2:expected1:phase2:difficulty2	-0.0117300	-0.0058574	0.0058726
diagnosis2:phase1	0.0141532	0.0117758	-0.0023774
diagnosis2:phase1:difficulty1	0.0137690	0.0044184	-0.0093506
diagnosis2:phase1:difficulty2	-0.0098694	-0.0054126	0.0044569
diagnosis2:phase2	0.0025945	0.0010793	-0.0015152
diagnosis2:phase2:difficulty1	0.0009623	0.0015290	0.0005667
diagnosis2:phase2:difficulty2	0.0057657	0.0030157	-0.0027500
diagnosis3	0.0147218	0.0165024	0.0017806
diagnosis3:difficulty1	0.0123957	0.0069778	-0.0054179
diagnosis3:difficulty2	-0.0250665	-0.0203600	0.0047065
diagnosis3:expected1	-0.0011207	-0.0041570	-0.0030364
diagnosis3:expected1:difficulty1	0.0091217	0.0132378	0.0041161
diagnosis3:expected1:difficulty2	-0.0017783	-0.0083723	-0.0065940
diagnosis3:expected1:phase1	-0.0031130	0.0053877	0.0085007
diagnosis3:expected1:phase1:difficulty1	0.0012655	0.0022258	0.0009604
diagnosis3:expected1:phase1:difficulty2	-0.0083082	-0.0048754	0.0034328
diagnosis3:expected1:phase2	0.0096356	0.0020228	-0.0076128
diagnosis3:expected1:phase2:difficulty1	-0.0071217	-0.0046146	0.0025071
diagnosis3:expected1:phase2:difficulty2	0.0061392	0.0032754	-0.0028638

fixed effect	use	cor	Estimate.diff
diagnosis3:phase1	-0.0024007	-0.0091461	-0.0067454
diagnosis3:phase1:difficulty1	0.0067041	0.0090339	0.0023298
diagnosis3:phase1:difficulty2	-0.0047899	-0.0119835	-0.0071937
diagnosis3:phase2	-0.0002878	0.0035756	0.0038634
diagnosis3:phase2:difficulty1	-0.0098692	-0.0082901	0.0015791
diagnosis3:phase2:difficulty2	0.0144615	0.0171624	0.0027010
difficulty1	-0.0431570	-0.0418193	0.0013377
difficulty2	-0.0102083	-0.0111177	-0.0009094
expected1	-0.0435172	-0.0509429	-0.0074256
expected1:difficulty1	-0.0234677	-0.0200669	0.0034008
expected1:difficulty2	0.0005227	-0.0000421	-0.0005648
expected1:phase1	-0.0387839	-0.0464322	-0.0076483
expected1:phase1:difficulty1	-0.0075735	-0.0095197	-0.0019463
expected1:phase1:difficulty2	-0.0023415	0.0000350	0.0023765
expected1:phase2	0.0000150	0.0016810	0.0016660
expected1:phase2:difficulty1	-0.0007777	-0.0018808	-0.0011031
expected1:phase2:difficulty2	-0.0087070	-0.0088048	-0.0000979
Intercept	5.9137150	5.9054759	-0.0082391
phase1	0.0297910	0.0301318	0.0003408
phase1:difficulty1	-0.0078720	-0.0025580	0.0053140
phase1:difficulty2	0.0232080	0.0249524	0.0017444
phase2	-0.0126536	-0.0124811	0.0001725
phase2:difficulty1	0.0025871	0.0028634	0.0002763
phase2:difficulty2	-0.0037692	-0.0063421	-0.0025729

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: 16 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 0.0450   "*"
##  2 ADHD      highneg_lowpos unexpected rt.cor       0.794 0.000402 "*"
##  3 ADHD      highpos_lowneg expected   rt.cor       0.874 0.00930  "*"
##  4 ADHD      highpos_lowneg unexpected rt.cor       0.904 0.0363   "*"
##  5 ASD       highneg_lowpos expected   rt.cor       0.883 0.0115   "*"
##  6 ASD       highneg_lowpos unexpected rt.cor       0.945 0.235    ""
##  7 ASD       highpos_lowneg expected   rt.cor       0.959 0.445    ""
##  8 ASD       highpos_lowneg unexpected rt.cor       0.886 0.0134   "*"
##  9 BOTH      highneg_lowpos expected   rt.cor       0.859 0.00395  "*"
## 10 BOTH      highneg_lowpos unexpected rt.cor       0.911 0.0421   "*"
## 11 BOTH      highpos_lowneg expected   rt.cor       0.842 0.00192  "*"
## 12 BOTH      highpos_lowneg unexpected rt.cor       0.884 0.0121   "*"
## 13 COMP      highneg_lowpos expected   rt.cor       0.928 0.109    ""
```

```
## 14 COMP      highneg_lowpos unexpected rt.cor       0.872 0.00861  "*"
## 15 COMP      highpos_lowneg expected   rt.cor       0.921 0.0784   ""
## 16 COMP      highpos_lowneg unexpected rt.cor       0.955 0.390    ""
```

```
# 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	10.458971	0.0148163	Wed Feb 26 08:51:31 2025	893f1a85749e
diagnosis + tone_pic + expected + tone_pic:expected	9.490657	0.0331604	Wed Feb 26 08:51:31 2025	893f88768a4
tone_pic + expected	9.080089	0.0084332	Wed Feb 26 08:51:31 2025	893f45014b8d
tone_pic + expected + tone_pic:expected	8.043344	0.0122361	Wed Feb 26 08:51:31 2025	893f12caf25
diagnosis + expected	7.438287	0.0378826	Wed Feb 26 08:51:31 2025	893f180d09ec
diagnosis + tone_pic + expected + diagnosis:expected	7.255477	0.1088670	Wed Feb 26 08:51:31 2025	893f10d605d5

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. [!Switch to something better????]

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

```
## # A tibble: 72 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 62 more rows
```

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

# run the ANOVA
if (!file.exists(file.path(brms_dir, "aov_acc.rds"))){
  aov.acc   = anovaBF(racc   ~ diagnosis * phase * expected * difficulty, data = df.acc)
  saveRDS(aov.acc, file = file.path(brms_dir, "aov_acc.rds"))
} else {
  aov.acc = readRDS(file.path(brms_dir, "aov_acc.rds"))
}

bf.acc = extractBF(aov.acc, logbf = T)
kable(head(bf.acc %>% arrange(desc(bf))))
```

	bf	error	time	code
diagnosis + expected + difficulty	61.99468	0.0147572	Fri Jan 31 13:34:33 2025	25e1017cb0c6f
diagnosis + difficulty	59.71508	0.0105499	Fri Jan 31 13:34:33 2025	25e102598c793
diagnosis + expected + difficulty + expected:difficulty	59.08581	0.0244052	Fri Jan 31 13:34:55 2025	25e1037d1b4f1
diagnosis + expected + difficulty + diagnosis:difficulty	58.71545	0.0165925	Fri Jan 31 13:34:36 2025	25e107f137933
expected + difficulty	57.67953	0.0113627	Fri Jan 31 13:34:33 2025	25e107eb14701
diagnosis + expected + diagnosis:expected + difficulty	57.66487	0.0372488	Fri Jan 31 13:34:33 2025	25e105446e94a

```
# 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: 4 x 3
##   diagnosis mean_accuracy sd_accuracy
##   <fct>             <dbl>       <dbl>
## 1 ADHD               90.2        12.8
## 2 ASD                88.8        14.1
## 3 BOTH               90.7        12.7
## 4 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        8.81
## 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                93.1        11.6
## 2 medium              92.6        11.5
## 3 difficult           86.3        14.0
```

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

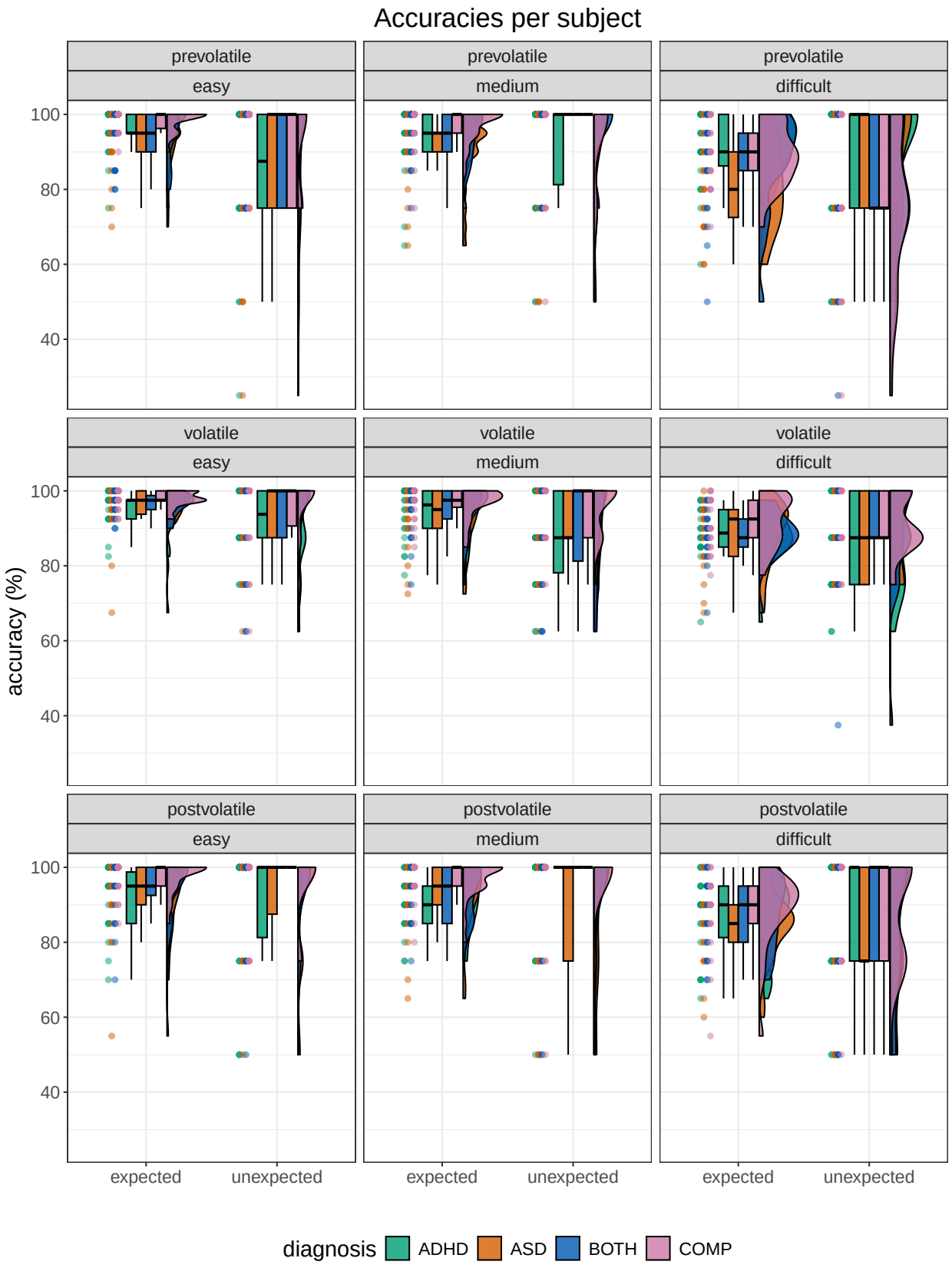
```
## # A tibble: 6 x 4
## # Groups:   expected [2]
##   expected   difficulty mean_accuracy sd_accuracy
##   <fct>      <fct>              <dbl>       <dbl>
## 1 expected   easy                95.1        6.82
## 2 expected   medium              93.7        7.67
## 3 expected   difficult           87.8        9.92
## 4 unexpected easy                91.1       14.7
## 5 unexpected medium              91.5       14.2
## 6 unexpected difficult           84.8       17.1
```

Accuracies were generally high, with a grand average of 90.66% accurate responses across diagnostic groups. Accuracies seems to have differed between diagnostic groups. Let's do some plotting to figure out where our differences lie.

Plots

```
# rain cloud plot including all factors

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 = .8),
violin.args = list(color = "black", outlier.shape = NA, alpha = .8),
boxplot.args.pos = list(
  position = ggpp::position_dodgenudge(x = 0, width = 0.3), width = 0.3
),
```

```
point.args = list(show_guide = FALSE, alpha = .5),
violin.args.pos = list(
  width = 0.6, position = position_nudge(x = 0.16)),
point.args.pos = list(position = ggpp::position_dodgenudge(x = -0.25, width = 0.1))) +
  scale_fill_manual(values = custom.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
BOTH
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) +
  ylim(75,100) +
  labs (x = "difficulty", y = "accuracy") +
  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))
```