ABN 50 multilevel - factors time, group, accuracy

Davide Gheza

09 August 2018

Load required packages:

```
if (!require("brms")) install.packages("brms")
library(brms)
theme_set(theme_default())

if (!require("tidyverse")) install.packages("tidyverse")
library(tidyverse)

if (!require("BEST")) install.packages("BEST")
library(BEST)

if (!require("ggplot2")) install.packages("ggplot2")
library(ggplot2)

if (!require("ggridges")) install.packages("ggridges")
library(ggridges)
```

Load data 50% reward probability, single trial, with time structure

This data comes from a reinfocement learning experiment. Subjects were instructed to learn, by trials and error, the stimulus-response association in 3 conditions: 100% (deterministic S-R), 80% (probabilistic), 50% (random). In each trial they saw a stimulus and they had to press 1 out of 2 possible buttons. They received a FB indicating the outcome (positive or negative). This experiment included two consecutive blocks of this task, adopting different stimuli (learning started from scratch in the first and second block). The trials belonging to the 3 conditions were intermixed, but here we selected only the random (50%) condition (where learning was made impossible by design, and thus there is a balanced amount of pos and neg FB). In this condition we expect an effect of time (slope) on FMT power, reflecting de-motivation (reduced cognitive control) in processing/exploring the FB. This data refer to "event related spectral perturbation" in the theta band (3-7Hz; FMT power, in dB) during the FB event.

After single-trial time/frequency decomposition and baseline normalization, for each accuracy condition (correct and incorrect FB) and task block (1st and 2nd), each single FB was combined with the time information, so that each single-trial FMT power was associated with the amount of time (rounded to seconds) elapsed from the beginning of each task block.

There were 2 theoretical approach to model this data: 1) consider the (simple) order of the single FB presentation 2) consider the exact time of the single FB presentation We chose the second, to improve the accuracy of the model (although in principle this equals to assuming a FB for each time point (second), and a lot of missing data).

Factors: group (HC vs MDD), accuracy (positive vs negative)

Continuous variable: time (sec)

not modelled factor -> block (1 vs 2)

This analysis aims to check whether the 2 groups (HC vs MDD) showed a different decay in FMT across time, and different for correct vs incorrect FB

Prepare the data

Load the data:

```
rm(list=ls())   # clear all!
wd <- ("C:/Users/gdavide/Documents/MATLAB/dg_workspace/Final_reanalysis_2017/results/HC vs MDDT1 - 34 s
setwd(wd) # set work directory
ABN50 <- read.csv2("50_34_HCvsMDDT1_fb_singletrials_ERSP_3_7_induced - with timeline.csv",header=TRUE,n

#head(ABN50)
```

Preparation of data we need

```
# long format
ABN50 = gather(ABN50, key = "tim", value = "power", X1:X800)


ABN50 = ABN50 %>% mutate(time = rep((1:length(unique(ABN50$tim))),          # rename time value as numbe
                                    each = length(ABN50$power)/800)) %>%
                  mutate(snG = paste0(ABN50$group, ABN50$sn)) %>%           # group-dependent coding for
                  filter(power != "0")                                      # exlcude missing data
                  #filter(blk == "blk2")                                     # (optional) select block
                  # filter(accuracy == "incorrect") %>%                      # (optional) select incorre

ABN50 = ABN50 %>%
        mutate(snG = as.factor(ABN50$snG))                                  # snG variable as factor

ABN50 = na.omit(ABN50)

# prepare conditions for marginal effects

conditions <- make_conditions(ABN50, c("accuracy", "group")) #, "blk"
```

Analyse the data: models definition and fitting

1) Fit a null model

(varying intercept and slope)

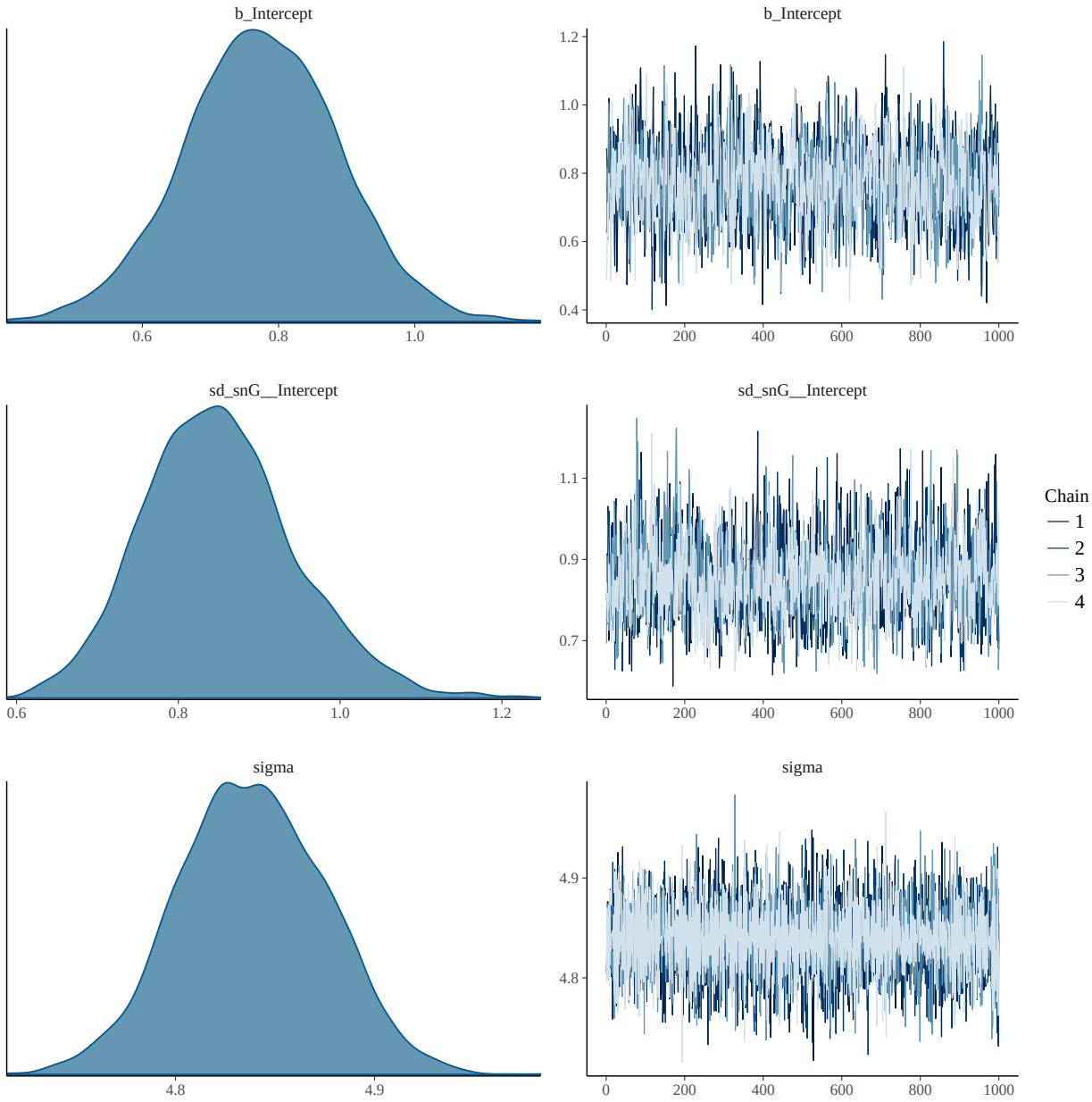
```
fit_null = readRDS(file = "ABN50_singletrial_timeline_accuracy_FITnull")

# fit_null <- brm(power ~ 1 + (1 | snG), data = ABN50, cores = 8,
#                     control = list(adapt_delta = 0.99))

# saveRDS(fit_null,file = "ABN50_singletrial_timeline_accuracy_FITnull")
```

```
summary(fit_null)
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: power ~ 1 + (1 | snG)
   Data: ABN50 (Number of observations: 8871)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Group-Level Effects:
```

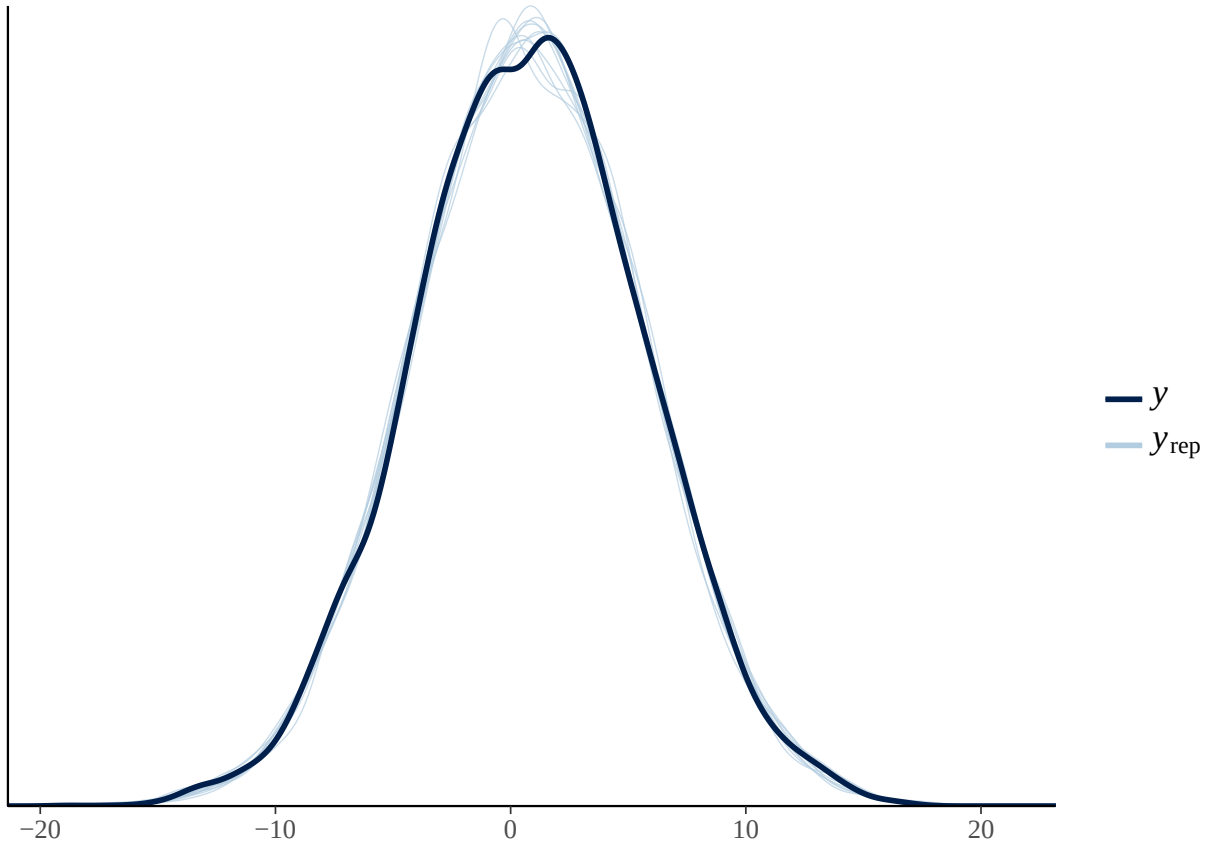
```
~snG (Number of levels: 68)
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)     0.85      0.09     0.68     1.05       1277 1.00

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept     0.77      0.11     0.55     1.00       1176 1.00

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma     4.84      0.04     4.77     4.91       4000 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
plot(fit_null, ask = FALSE)
```

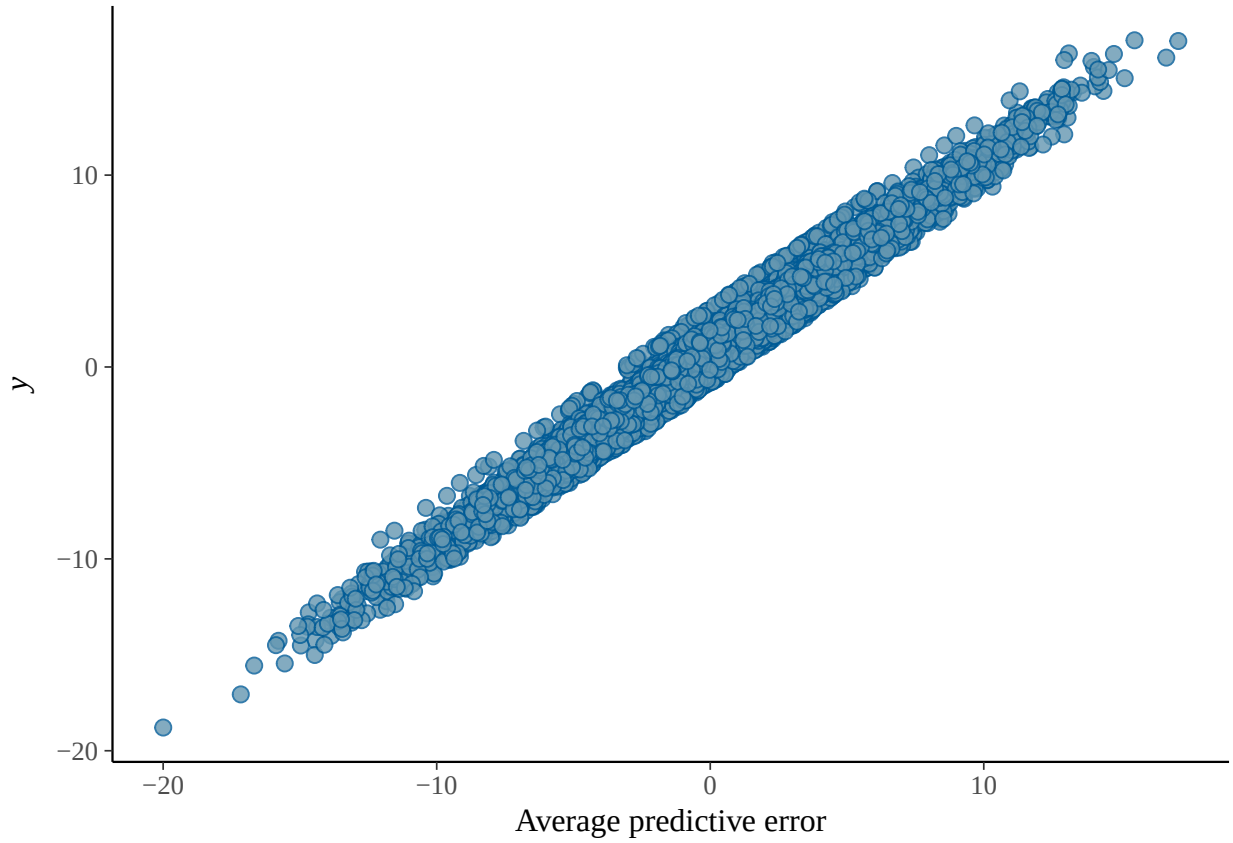


Check model fit:

```
pp_check(fit_null)
```




```
pp_check(fit_null, "error_scatter_avg")
```



2) Fit a simple model

(main effect of time - varying intercept and slope)

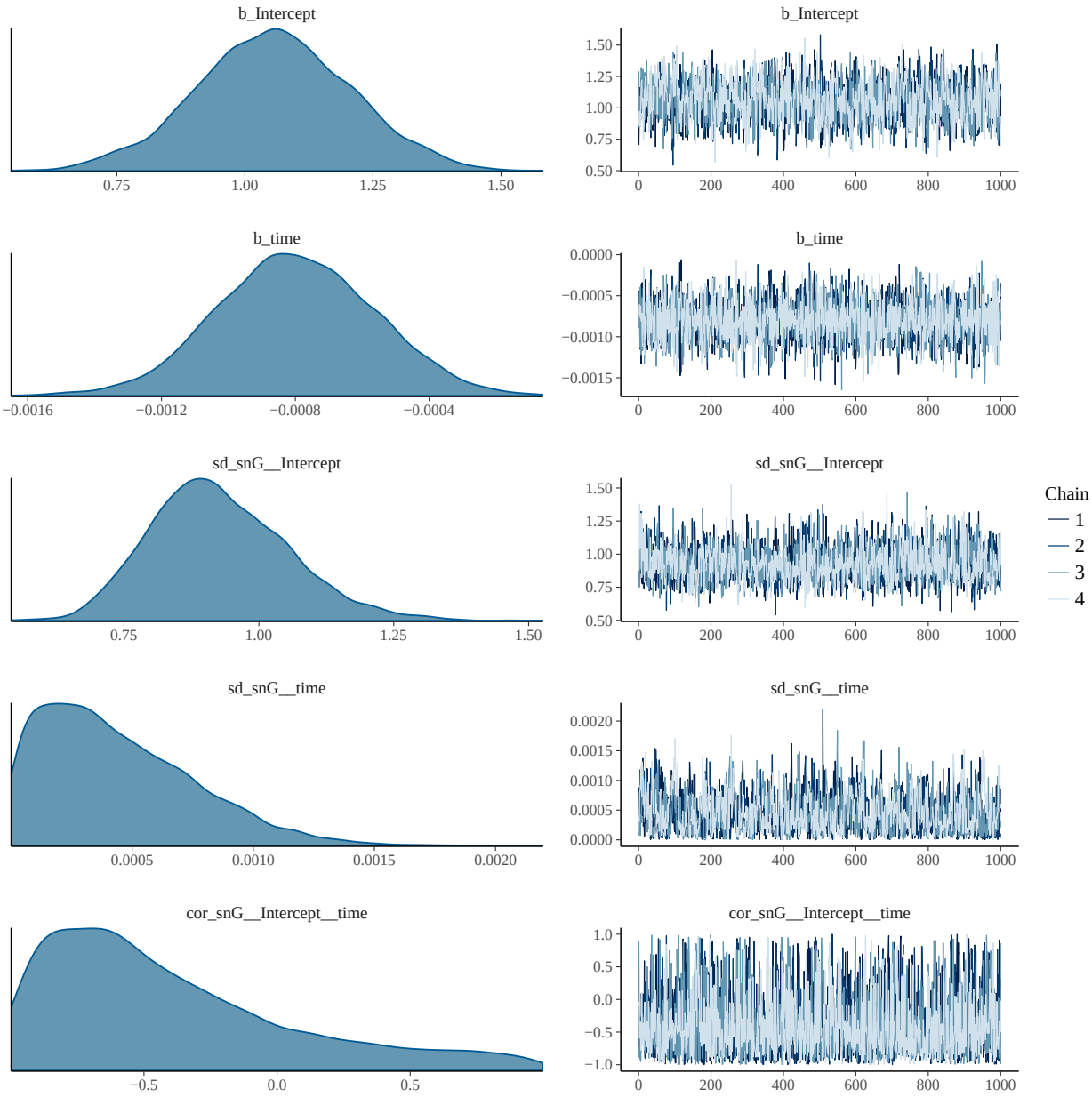
```
fit_compact = readRDS(file = "ABN50_singletrial_timeline_accuracy_FITcompact")

# fit_compact <- brm(power ~ 1 + time + (1 + time | snG), data = ABN50, cores = 8,
#                    control = list(adapt_delta = 0.99))

# saveRDS(fit_compact,file = "ABN50_singletrial_timeline_accuracy_FITcompact")
```

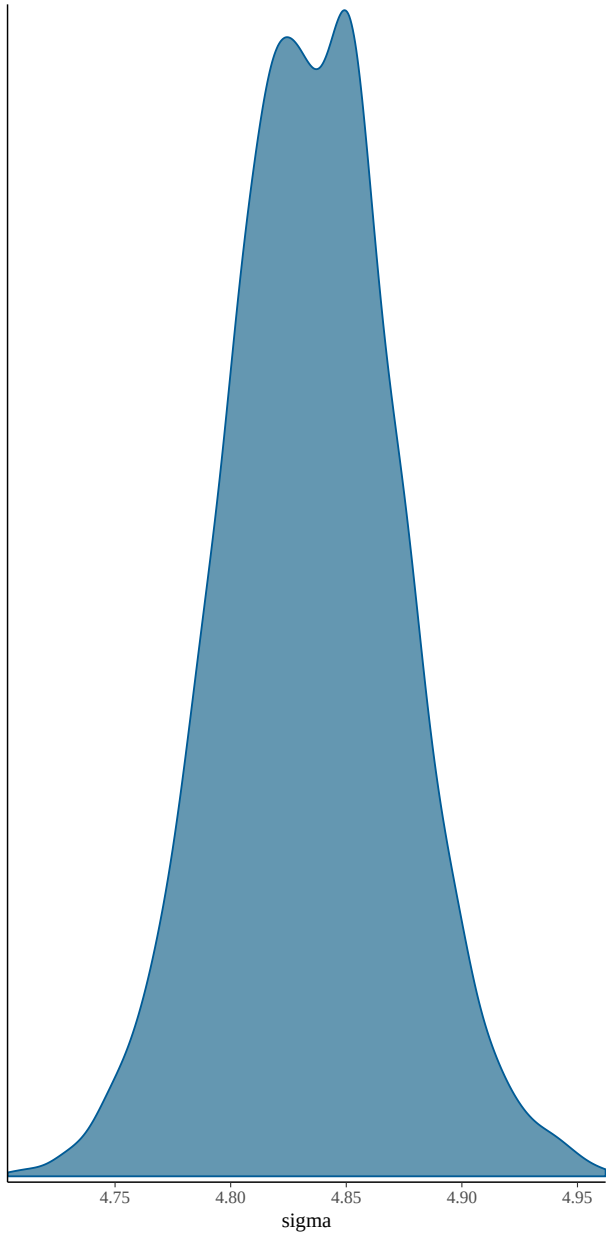
```
summary(fit_compact)
```

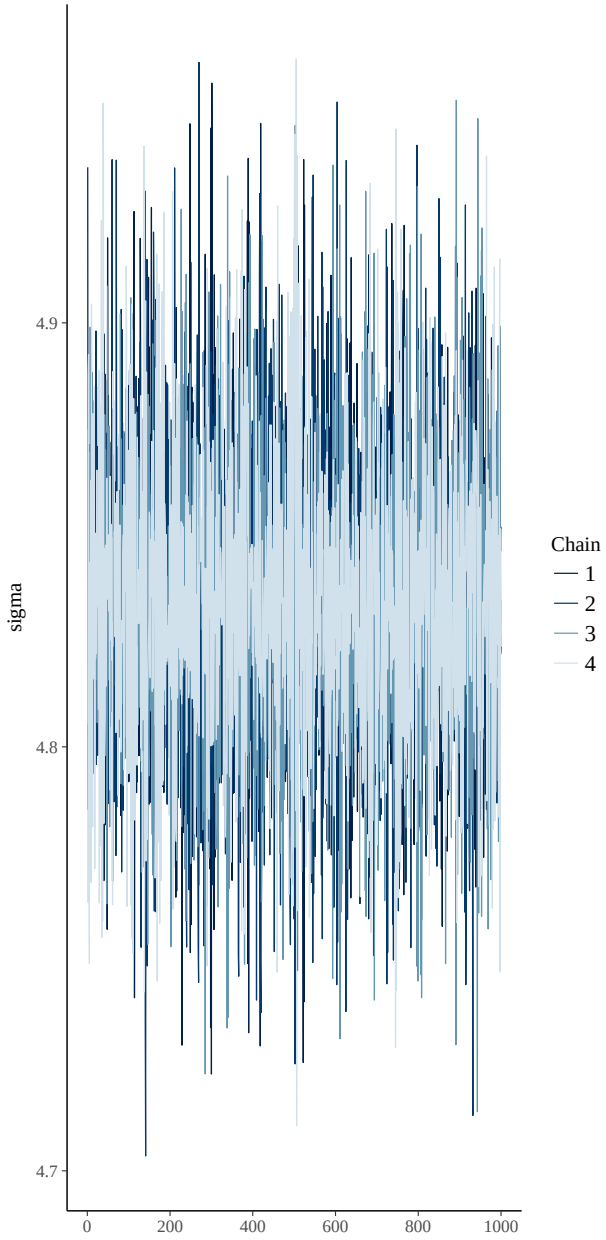
```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: power ~ 1 + time + (1 + time | snG)
   Data: ABN50 (Number of observations: 8871)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Group-Level Effects:
~snG (Number of levels: 68)
                  Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)         0.93      0.13     0.71     1.21       1477 1.00
sd(time)              0.00      0.00     0.00     0.00        733 1.01
cor(Intercept,time)  -0.40      0.47    -0.97     0.78       3066 1.00
```

```
Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept     1.06      0.15     0.75     1.35       1598 1.00
time         -0.00      0.00    -0.00    -0.00       4000 1.00

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma     4.83      0.04     4.76     4.91       4000 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot(fit_compact, ask = FALSE)
```



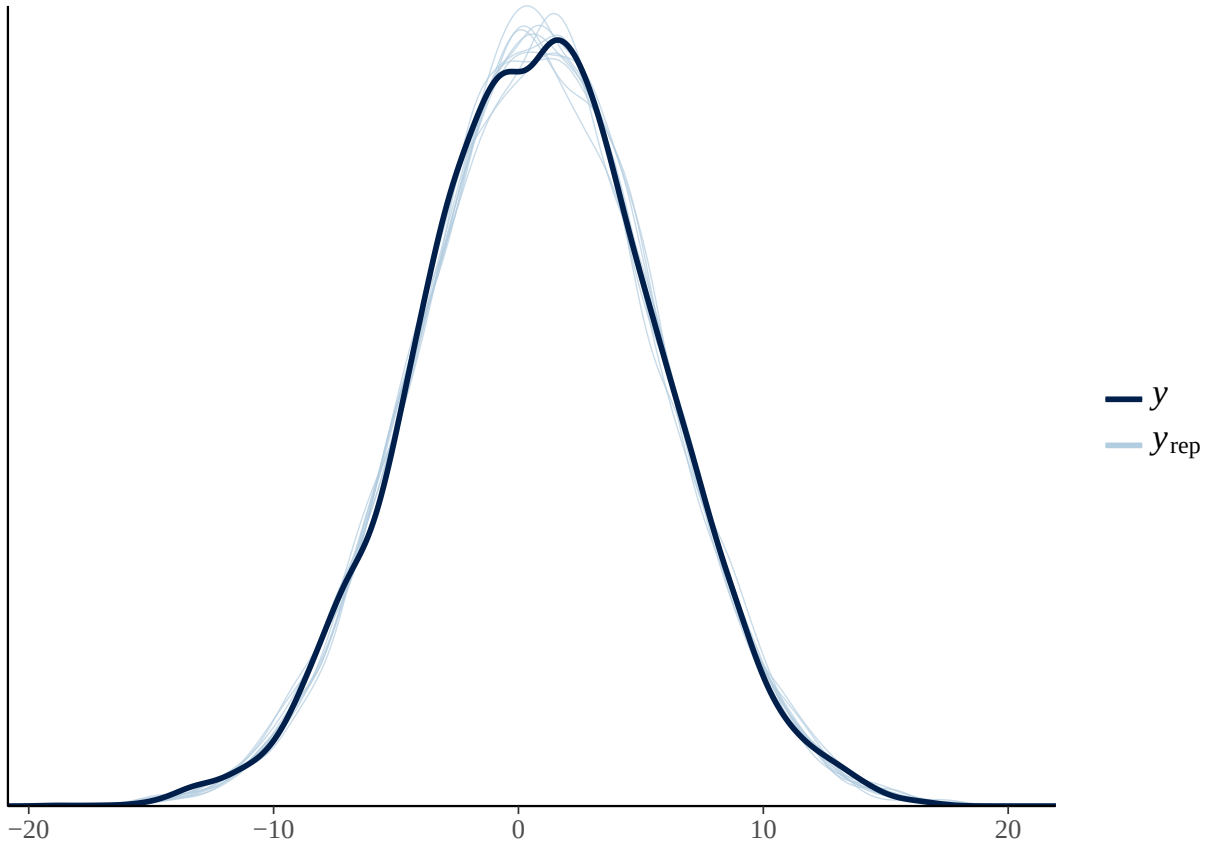




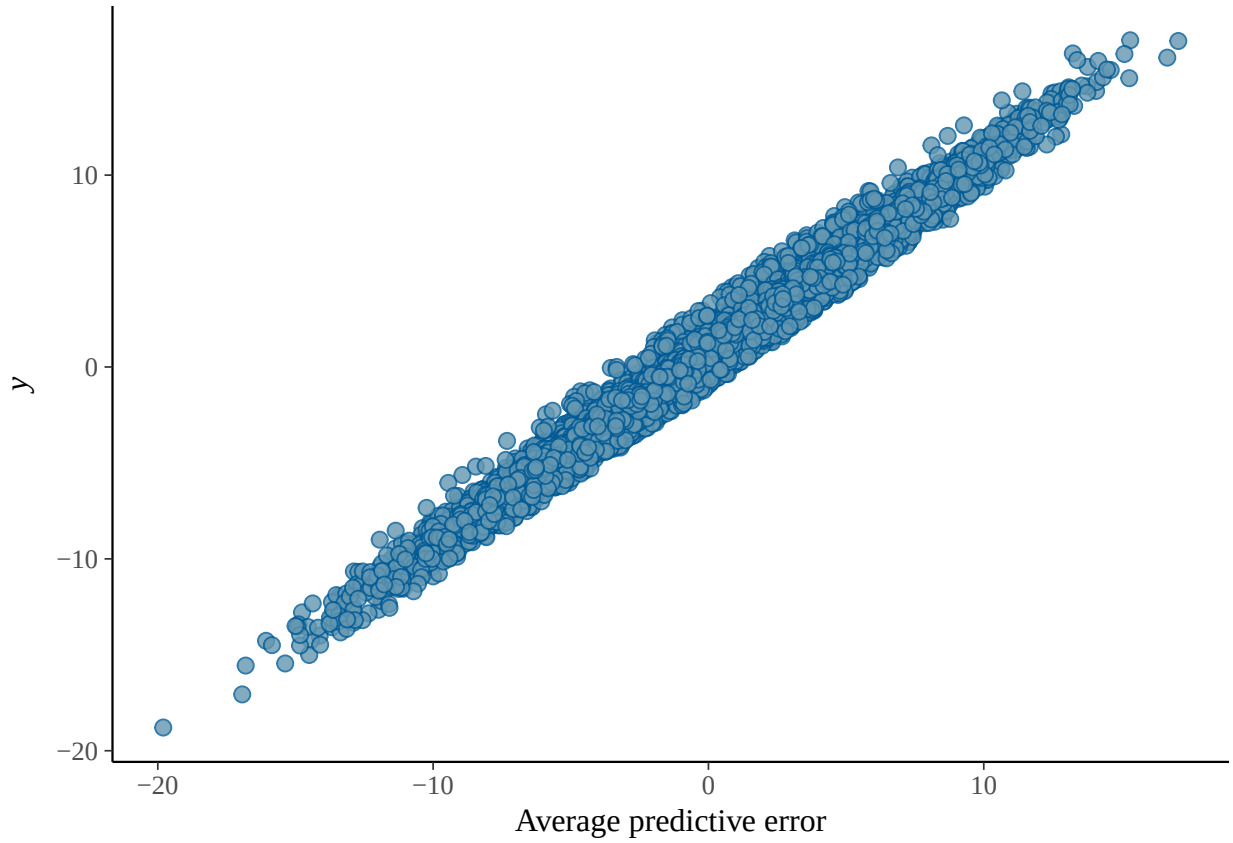
Visualize predictions:

Check model fit:

```
pp_check(fit_compact)
```




```
pp_check(fit_compact, "error_scatter_avg")
```



3) Fit an extended model - time*accuracy (main effects + interaction time and accuracy - varying intercept, main effects + interaction)

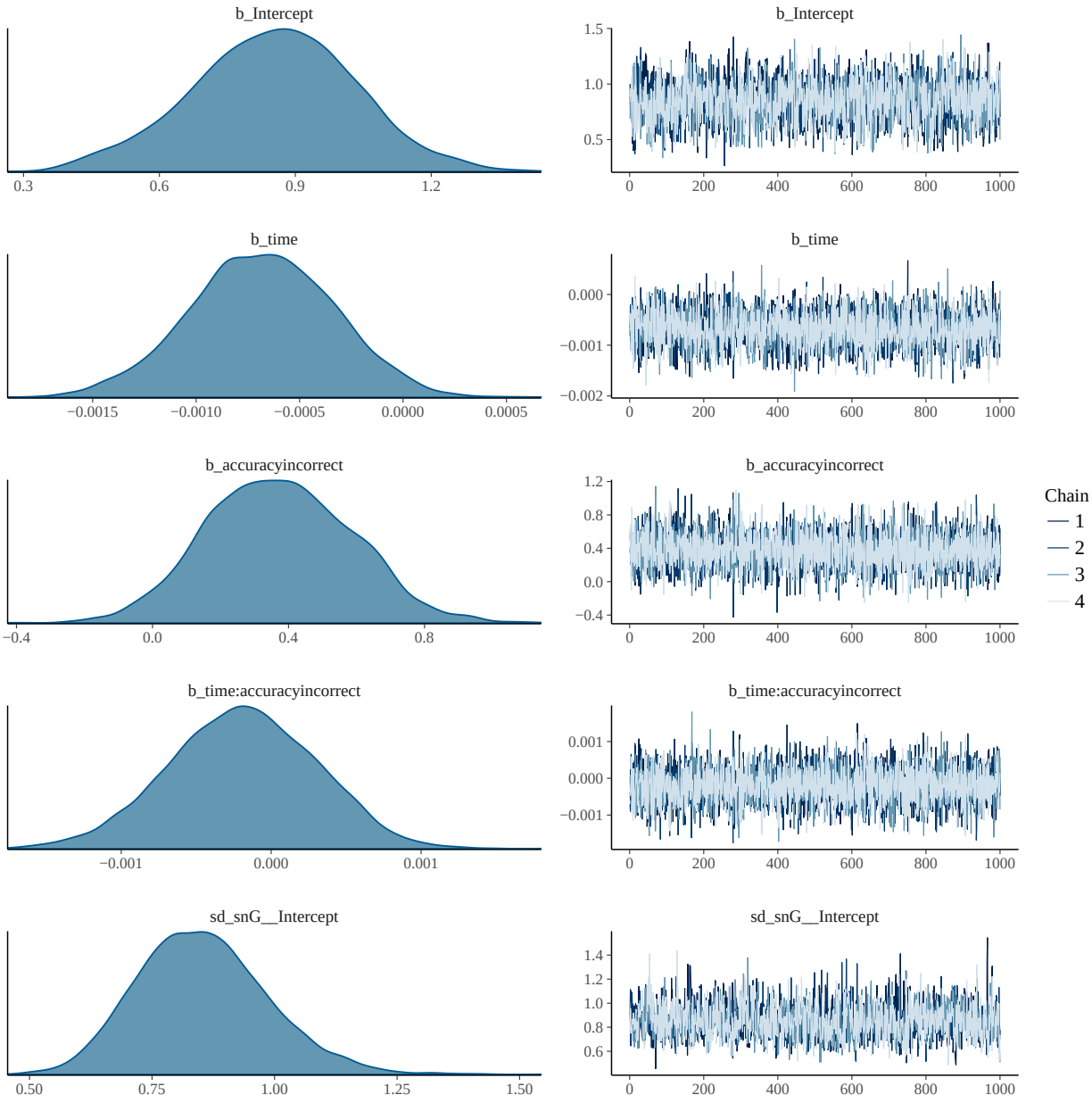
```
fit_timeaccuracy = readRDS(file = "ABN50_singletrial_timeline_accuracy_FITtimeaccuracy")

# fit_timeaccuracy <- brm(power ~ 1 + time*accuracy + (1 + time*accuracy | snG), data = ABN50, cores = 
#                     control = list(adapt_delta = 0.99))

# saveRDS(fit_timeaccuracy,file = "ABN50_singletrial_timeline_accuracy_FITtimeaccuracy")
```

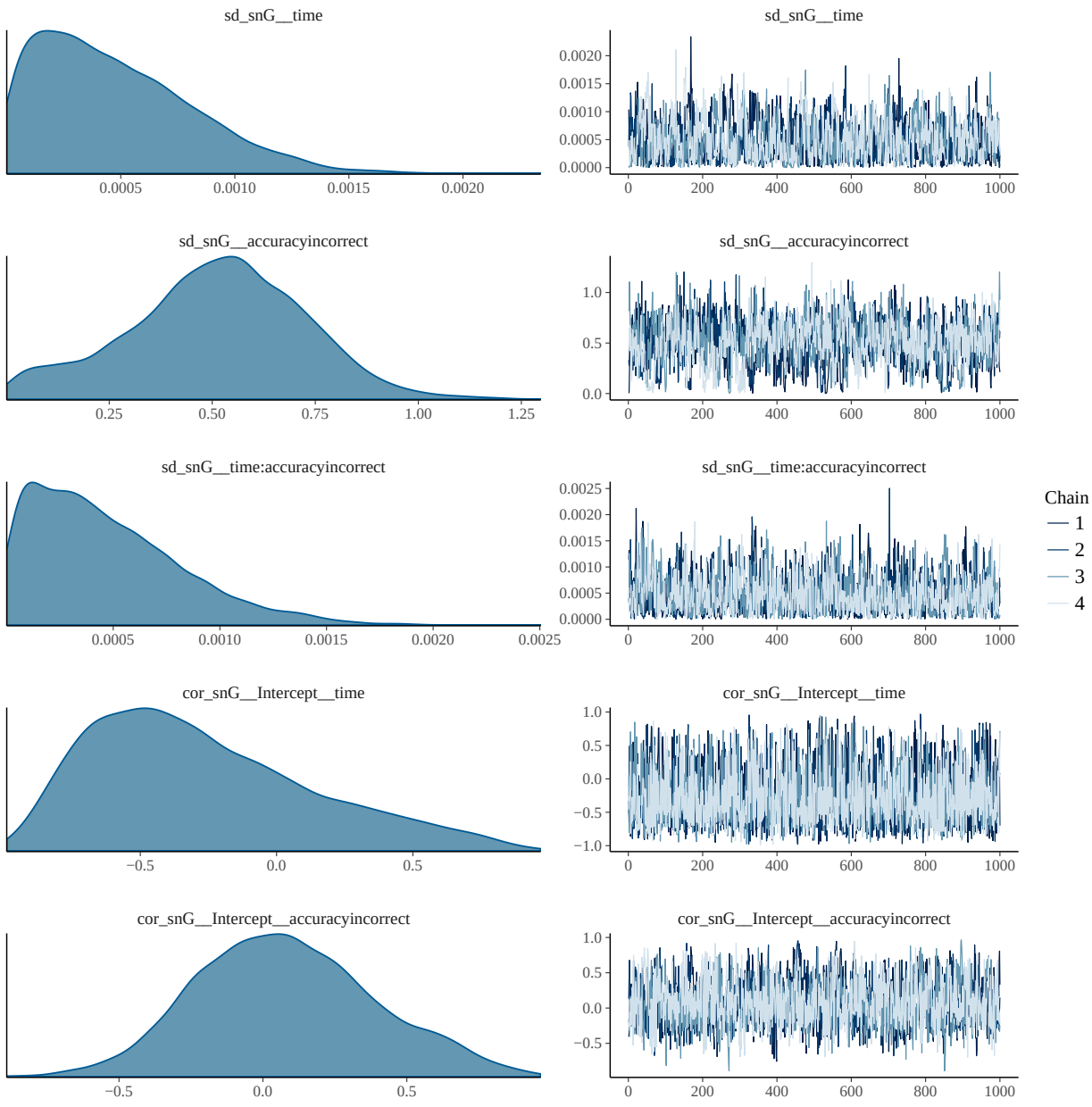
```
summary(fit_timeaccuracy)
```

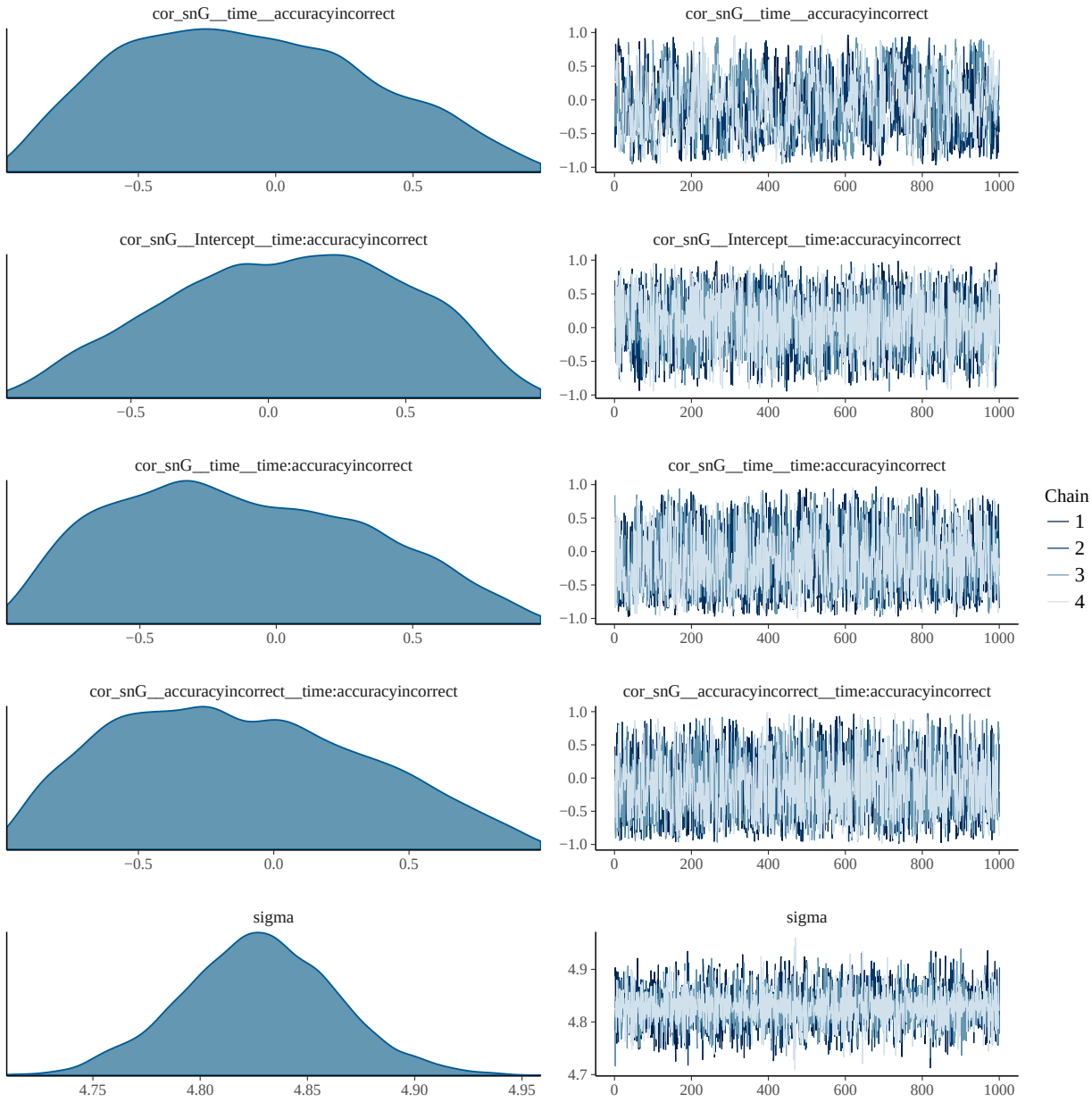
```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: power ~ 1 + time * accuracy + (1 + time * accuracy | snG)
   Data: ABN50 (Number of observations: 8871)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Group-Level Effects:
~snG (Number of levels: 68)
                                    Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)                           0.85      0.13     0.62     1.14       1796 1.00
sd(time)                                0.00      0.00     0.00     0.00        633 1.01
sd(accuracyincorrect)                   0.51      0.21     0.06     0.91        467 1.00
sd(time:accuracyincorrect)              0.00      0.00     0.00     0.00       1087 1.00
```

```
cor(Intercept,time)                                -0.27      0.41    -0.86     0.66       2021 1.00
cor(Intercept,accuracyincorrect)                    0.08      0.31    -0.49     0.72       1446 1.00
cor(time,accuracyincorrect)                        -0.10      0.44    -0.84     0.76        386 1.02
cor(Intercept,time:accuracyincorrect)               0.07      0.42    -0.74     0.81       4000 1.00
cor(time,time:accuracyincorrect)                   -0.13      0.45    -0.86     0.76       1700 1.00
cor(accuracyincorrect,time:accuracyincorrect)      -0.13      0.45    -0.87     0.77       2405 1.00

Population-Level Effects:
                      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept                 0.85      0.18     0.49     1.20       2034 1.00
time                     -0.00      0.00    -0.00    -0.00       4000 1.00
accuracyincorrect         0.37      0.21    -0.03     0.79       4000 1.00
time:accuracyincorrect   -0.00      0.00    -0.00     0.00       4000 1.00

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma     4.83      0.04     4.76     4.90       4000 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
plot(fit_timeaccuracy, ask = FALSE)
```


b_Intercept
b_Intercept
b_time
b_time
b_accuracyincorrect
b_accuracyincorrect
b_time:accuracyincorrect
b_time:accuracyincorrect
sd_snG__Intercept
sd_snG__Intercept
Chain
1
2
3
4

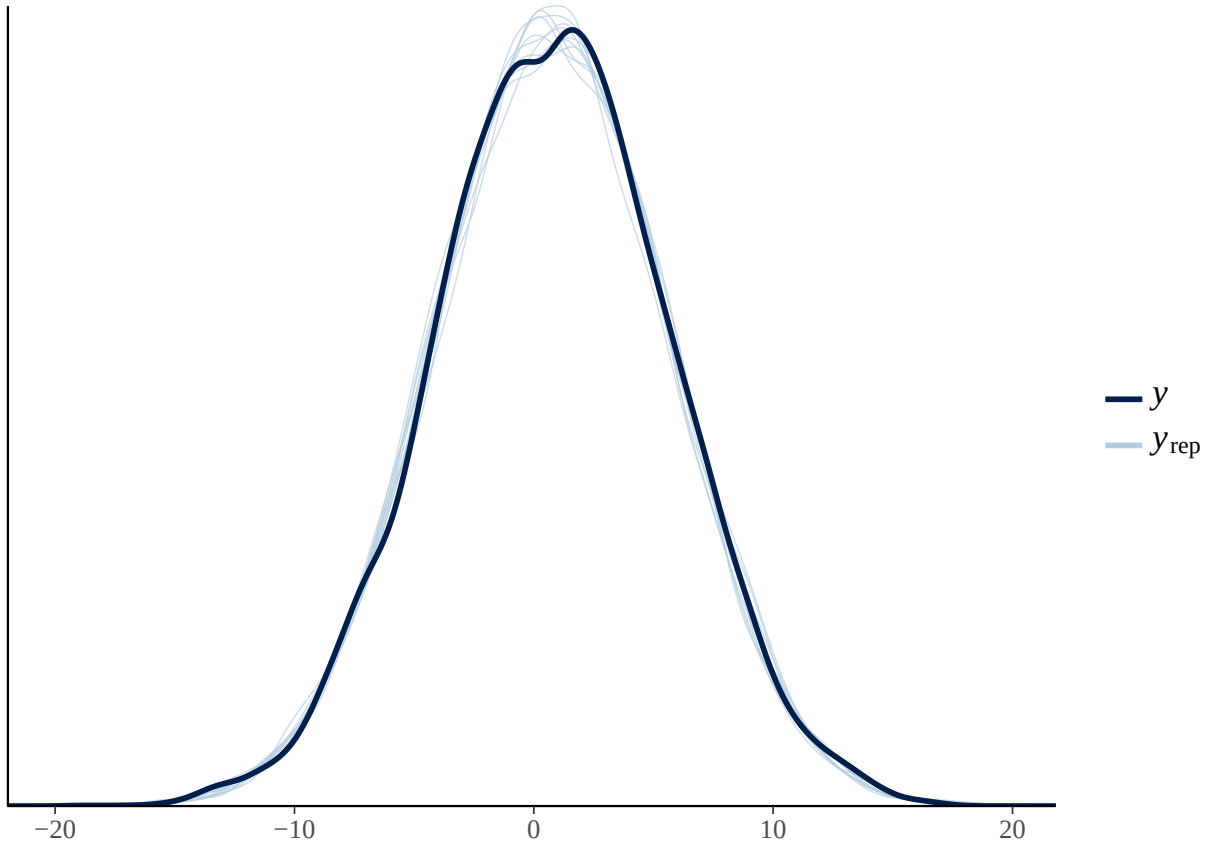

sd_snG__time
0.0005
0.0010
0.0015
0.0020
sd_snG__time
0.0020
0.0015
0.0010
0.0005
0.0000
0
200
400
600
800
1000
sd_snG__accuracyincorrect
0.25
0.50
0.75
1.00
1.25
sd_snG__accuracyincorrect
1.0
0.5
0.0
sd_snG__time:accuracyincorrect
0.0005
0.0010
0.0015
0.0020
0.0025
sd_snG__time:accuracyincorrect
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
Chain
1
2
3
4
cor_snG__Intercept__time
−0.5
0.0
0.5
cor_snG__Intercept__time
1.0
0.5
0.0
−0.5
−1.0
cor_snG__Intercept__accuracyincorrect
cor_snG__Intercept__accuracyincorrect



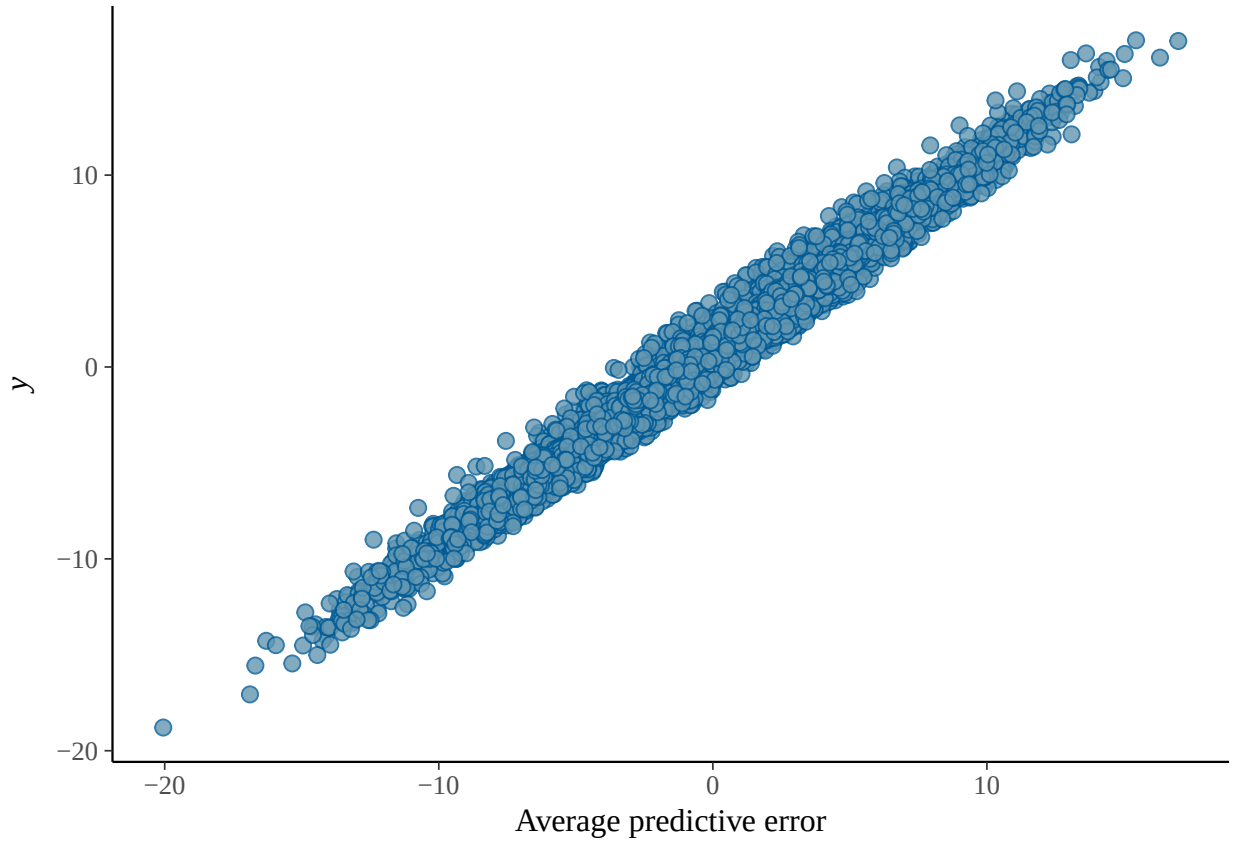
Visualize predictions:

Check model fit:

```
pp_check(fit_timeaccuracy)
```




```
pp_check(fit_timeaccuracy, "error_scatter_avg")
```



4) Fit an extended model - time*group (main effects + interaction time and group - varying intercept and slope)

```
fit_timegroup = readRDS(file = "ABN50_singletrial_timeline_accuracy_FITtimegroup")

#Ladislas: between deisgn: group should not be set as varying effect
# fit_timegroup <- brm(power ~ 1 + time*group + (1 + time | snG), data = ABN50, cores = 8,
#                  control = list(adapt_delta = 0.99))

# # (Antonio's) fit_full <- brm(power ~ 1 + time*group + (1 + time*group | snG), data = ABN50, cores =
#                 control = list(adapt_delta = 0.99))

# saveRDS(fit_timegroup,file = "ABN50_singletrial_timeline_accuracy_FITtimegroup")
```

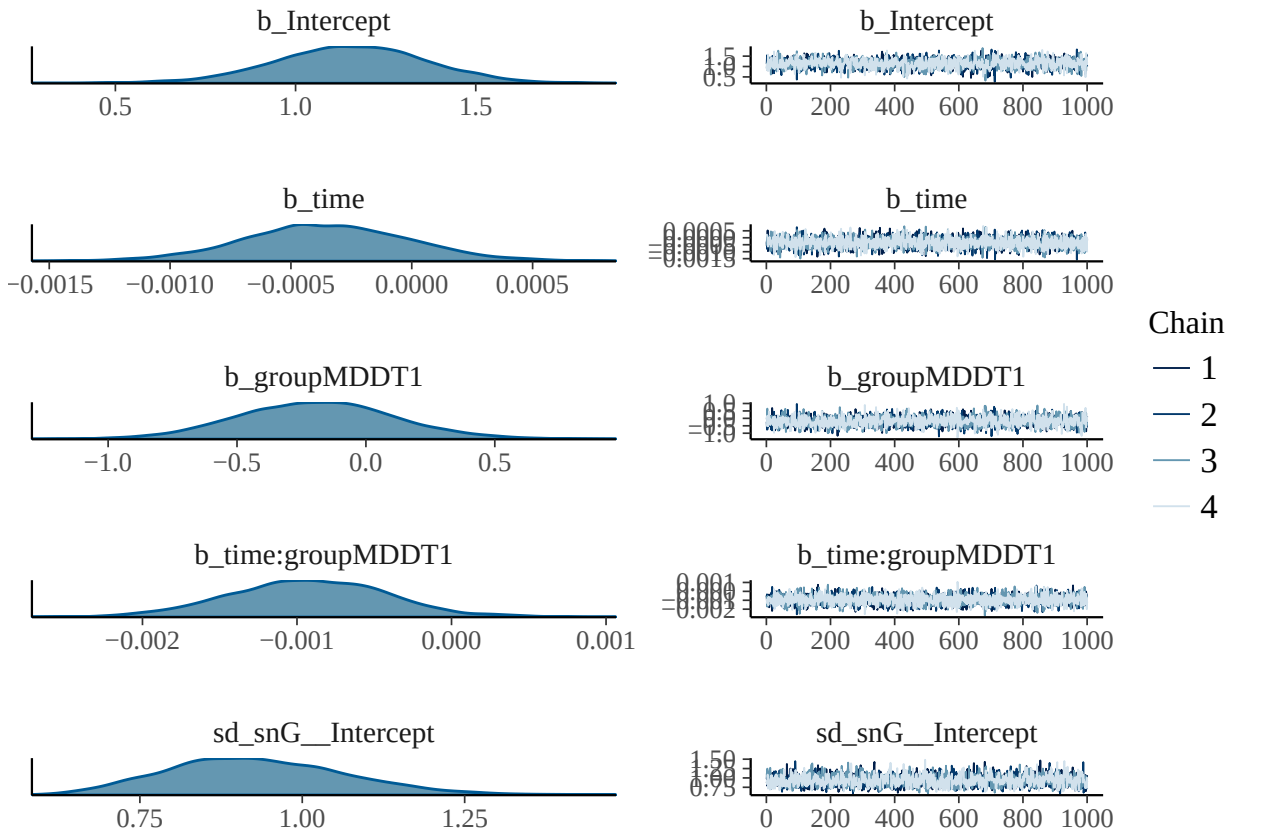
Summarize results:

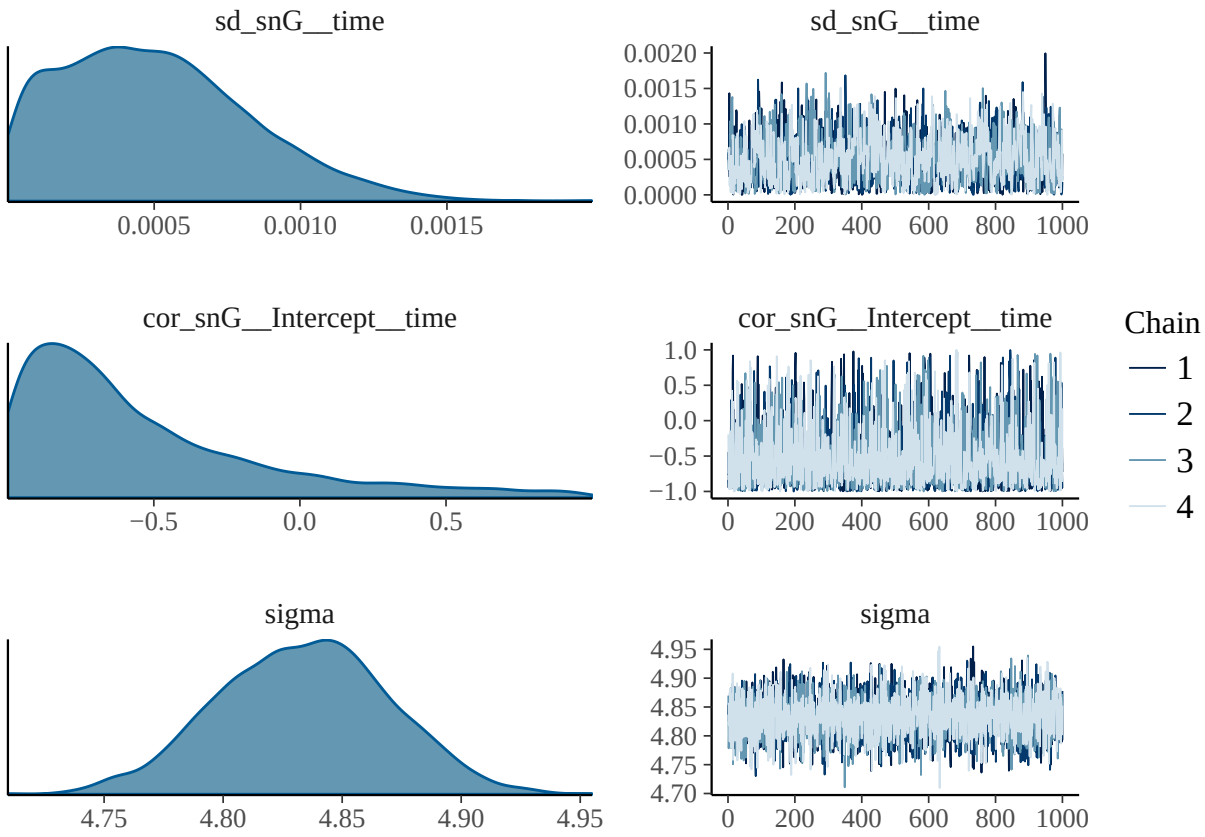
```
summary(fit_timegroup)
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: power ~ 1 + time * group + (1 + time | snG)
   Data: ABN50 (Number of observations: 8871)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Group-Level Effects:
```

```
~snG (Number of levels: 68)
                    Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)           0.93      0.14     0.69     1.22       1415 1.00
sd(time)                0.00      0.00     0.00     0.00        920 1.00
cor(Intercept,time)    -0.54      0.42    -0.99     0.65       2183 1.00

Population-Level Effects:
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept         1.15      0.21     0.73     1.55       2230 1.00
time             -0.00      0.00    -0.00     0.00       4000 1.00
groupMDDT1       -0.19      0.31    -0.78     0.43       2396 1.00
time:groupMDDT1  -0.00      0.00    -0.00     0.00       4000 1.00

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma     4.83      0.04     4.76     4.90       4000 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
plot(fit_timegroup, ask = FALSE)
```





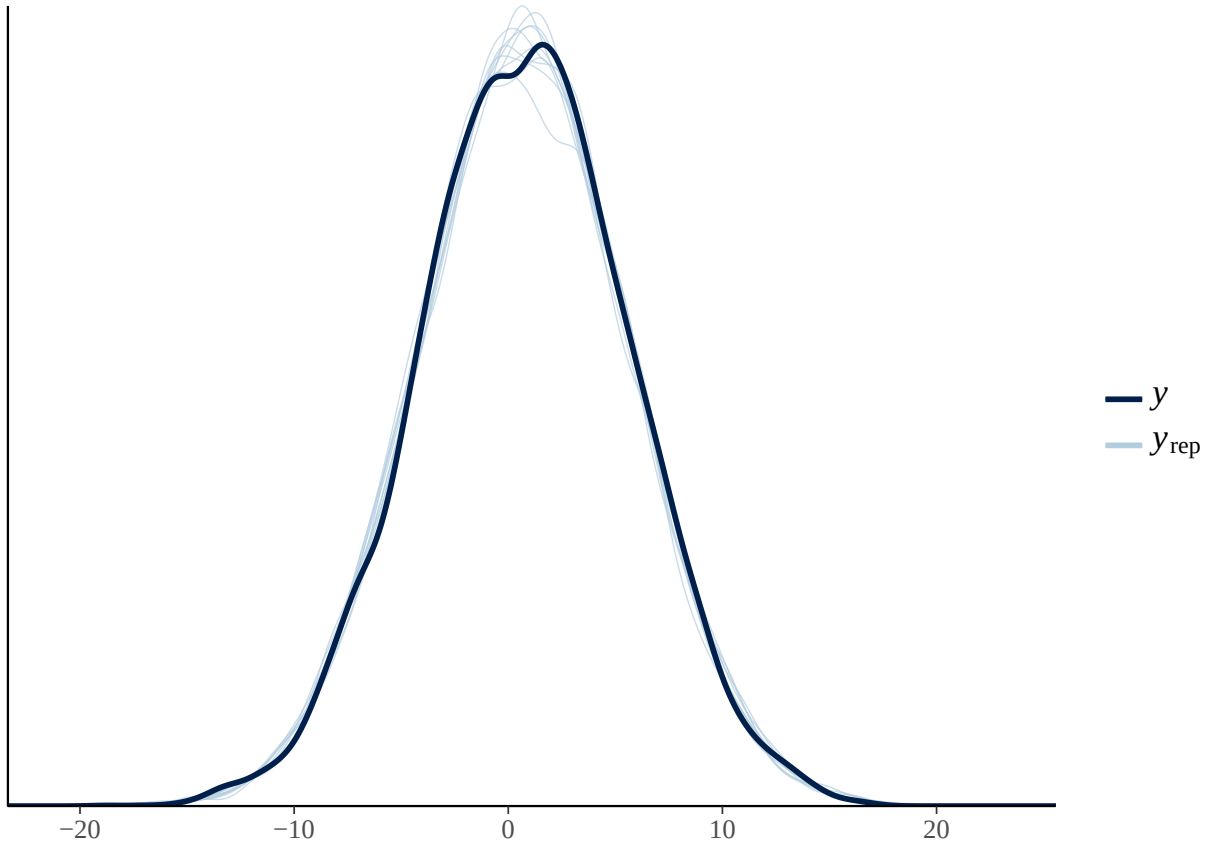
Vizualize predictions:

Compute evidence ratio for slope HC vs MDD

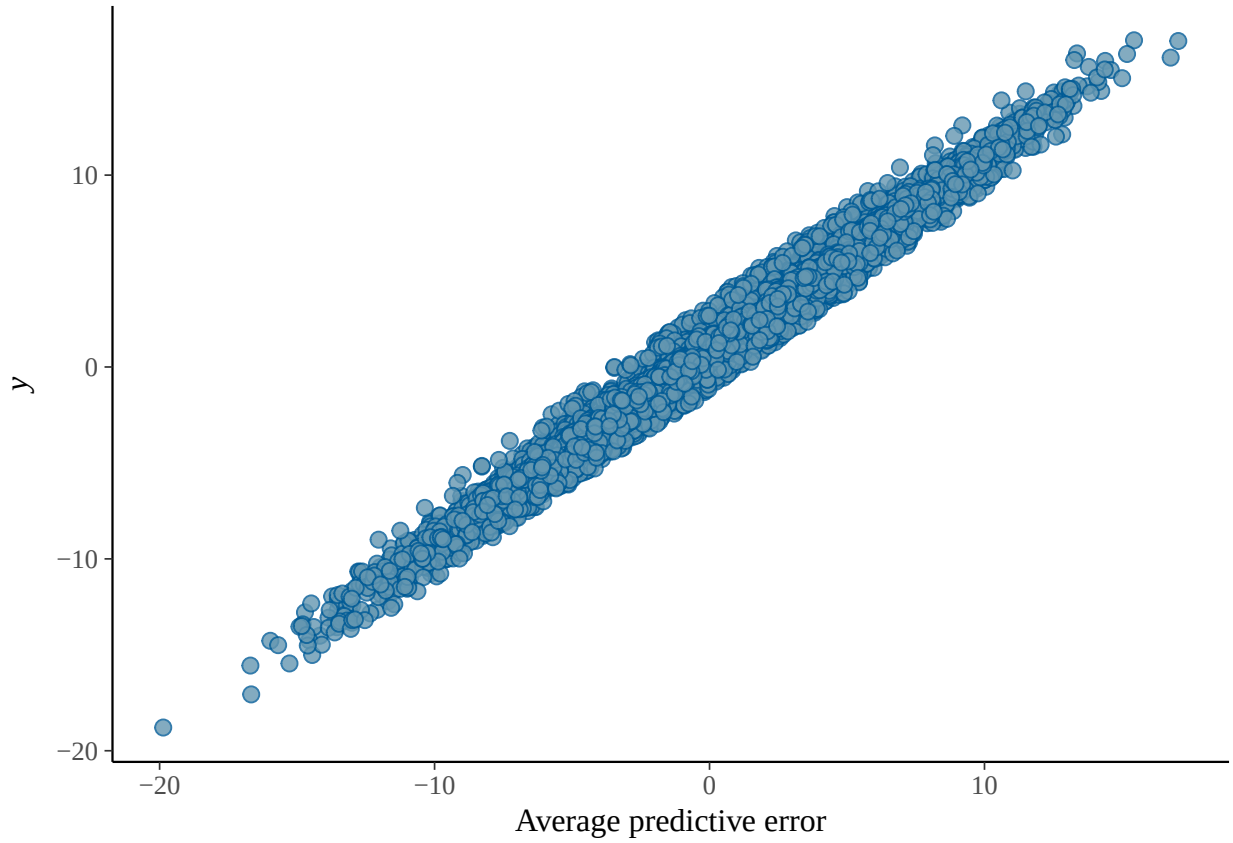
Compute fitted values and residuals

Check model fit:

```
pp_check(fit_timegroup)
```




```
pp_check(fit_timegroup, "error_scatter_avg")
```



5) Fit an extended model - time * accuracy + time * group

(main effects + interactions of time * accuracy and time * group - varying intercept, main effects + interactions)

```
fit_timeaccuracytimegroup = readRDS(file = "ABN50_singletrial_timeline_accuracy_FITtimeaccuracytimegrou

#Ladislas: between design: group should not be set as varying effect
# fit_timeaccuracytimegroup <- brm(power ~ 1 + time*accuracy + group + time:group + (1 + time*accuracy
#                                  cores = 8, control = list(adapt_delta = 0.99))


# saveRDS(fit_timeaccuracytimegroup,file = "ABN50_singletrial_timeline_accuracy_FITtimeaccuracytimegrou
```

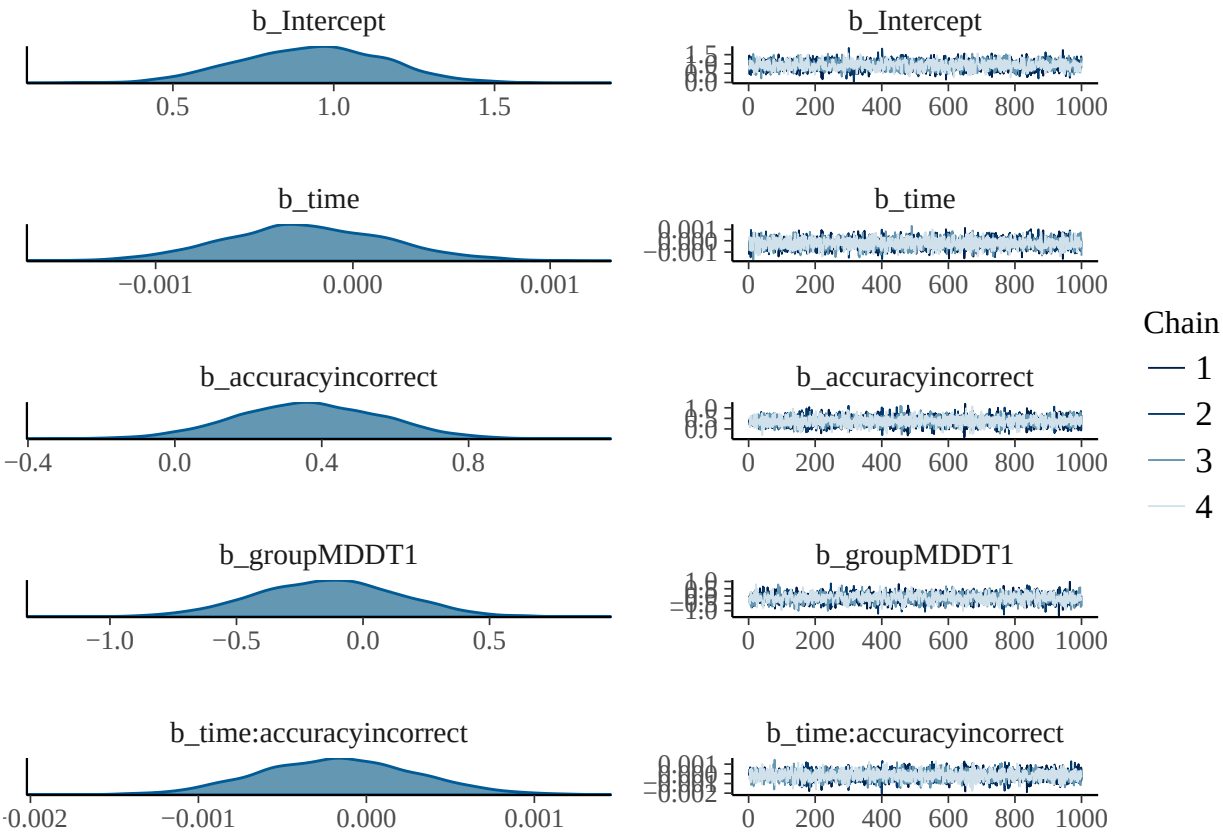
Summarize results:

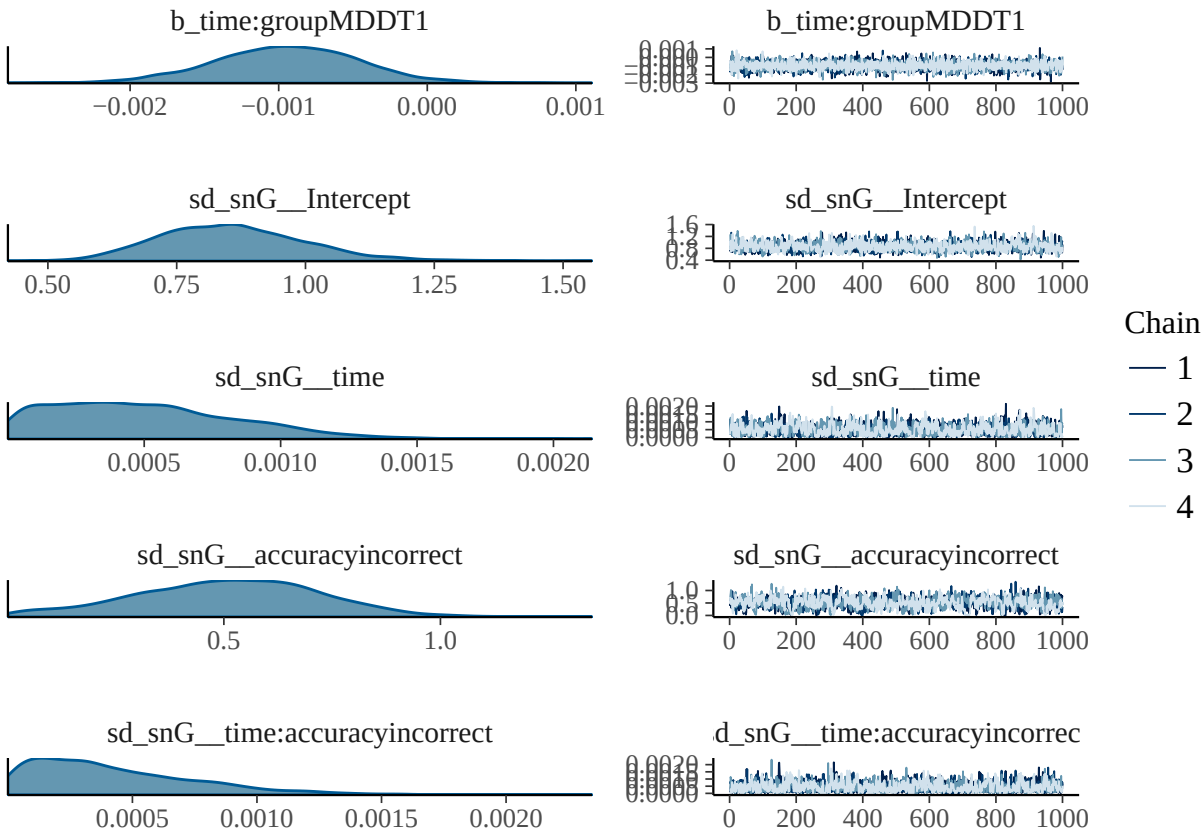
```
summary(fit_timeaccuracytimegroup)
```

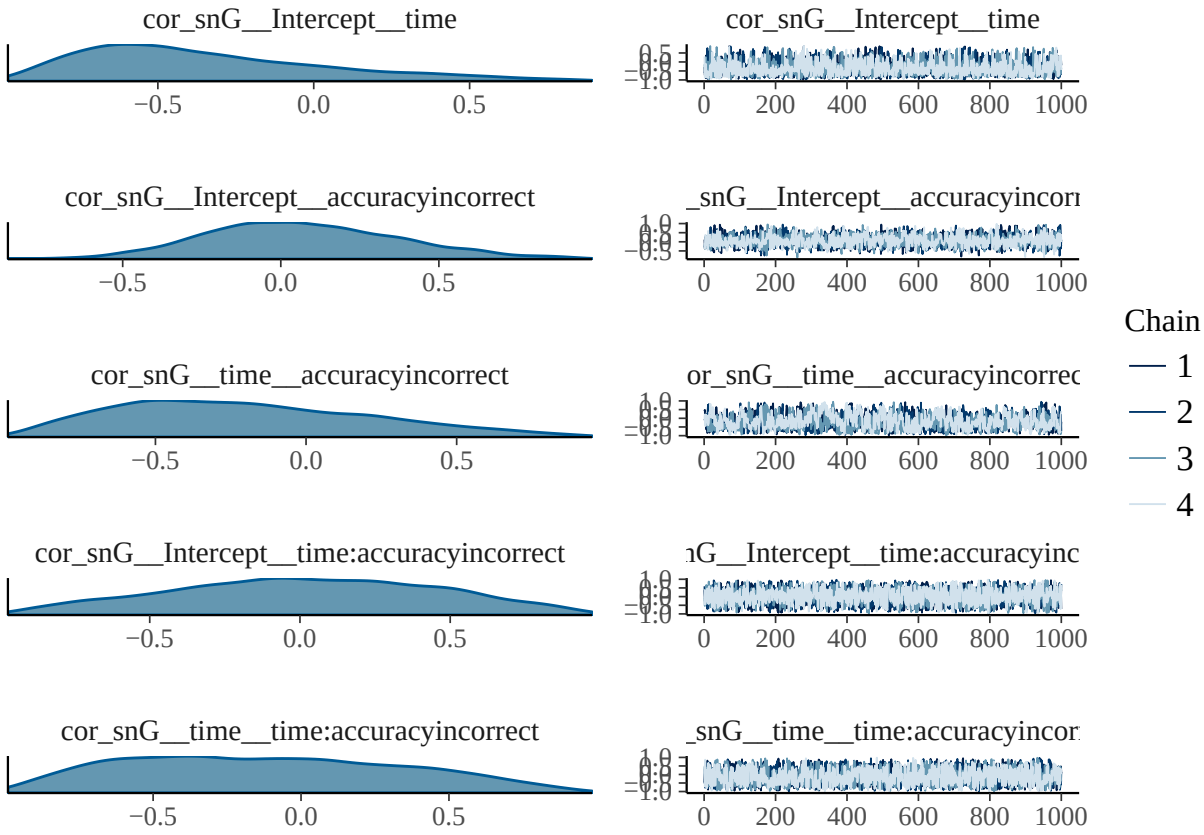
```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: power ~ 1 + time * accuracy + group + time:group + (1 + time * accuracy | snG)
   Data: ABN50 (Number of observations: 8871)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Group-Level Effects:
~snG (Number of levels: 68)
                                                Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
```

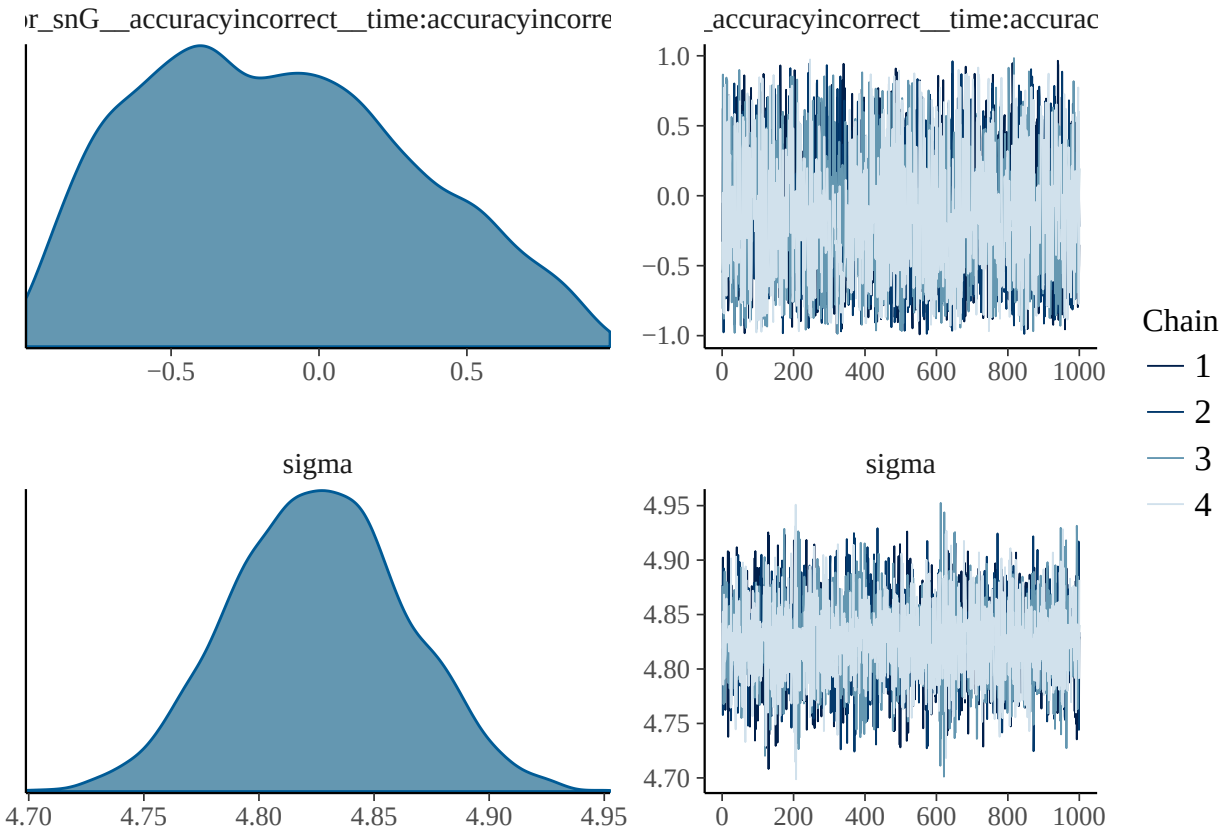
```
sd(Intercept)                                    0.86      0.14     0.61     1.17       1842 1.00
sd(time)                                         0.00      0.00     0.00     0.00        915 1.00
sd(accuracyincorrect)                            0.51      0.21     0.07     0.90        546 1.01
sd(time:accuracyincorrect)                       0.00      0.00     0.00     0.00       1277 1.00
cor(Intercept,time)                             -0.34      0.39    -0.90     0.58       4000 1.00
cor(Intercept,accuracyincorrect)                 0.07      0.31    -0.49     0.71       2034 1.00
cor(time,accuracyincorrect)                     -0.20      0.42    -0.86     0.69        391 1.01
cor(Intercept,time:accuracyincorrect)            0.02      0.44    -0.82     0.82       4000 1.00
cor(time,time:accuracyincorrect)                -0.12      0.46    -0.88     0.76       4000 1.00
cor(accuracyincorrect,time:accuracyincorrect)   -0.14      0.46    -0.88     0.78       4000 1.00

Population-Level Effects:
                     Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept                0.93      0.23     0.49     1.37       2855 1.00
time                    -0.00      0.00    -0.00     0.00       4000 1.00
accuracyincorrect        0.37      0.21    -0.02     0.77       4000 1.00
groupMDDT1              -0.14      0.29    -0.70     0.41       2389 1.00
time:accuracyincorrect  -0.00      0.00    -0.00     0.00       4000 1.00
time:groupMDDT1         -0.00      0.00    -0.00     0.00       4000 1.00

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma     4.83      0.04     4.75     4.90       4000 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot(fit_timeaccuracytimegroup, ask = FALSE)
```


b_Intercept
b_Intercept
b_time
b_time
b_accuracyincorrect
b_accuracyincorrect
b_groupMDDT1
b_groupMDDT1
b_time:accuracyincorrect
b_time:accuracyincorrect
Chain
1
2
3
4


b_time:groupMDDT1
−0.002
−0.001
0.000
0.001
b_time:groupMDDT1
0
200
400
600
800
1000
sd_snG__Intercept
0.50
0.75
1.00
1.25
1.50
sd_snG__Intercept
0
200
400
600
800
1000
sd_snG__time
0.0005
0.0010
0.0015
0.0020
sd_snG__time
0
200
400
600
800
1000
sd_snG__accuracyincorrect
0.5
1.0
sd_snG__accuracyincorrect
0
200
400
600
800
1000
sd_snG__time:accuracyincorrect
0.0005
0.0010
0.0015
0.0020
d_snG__time:accuracyincorrec
0
200
400
600
800
1000
Chain
1
2
3
4

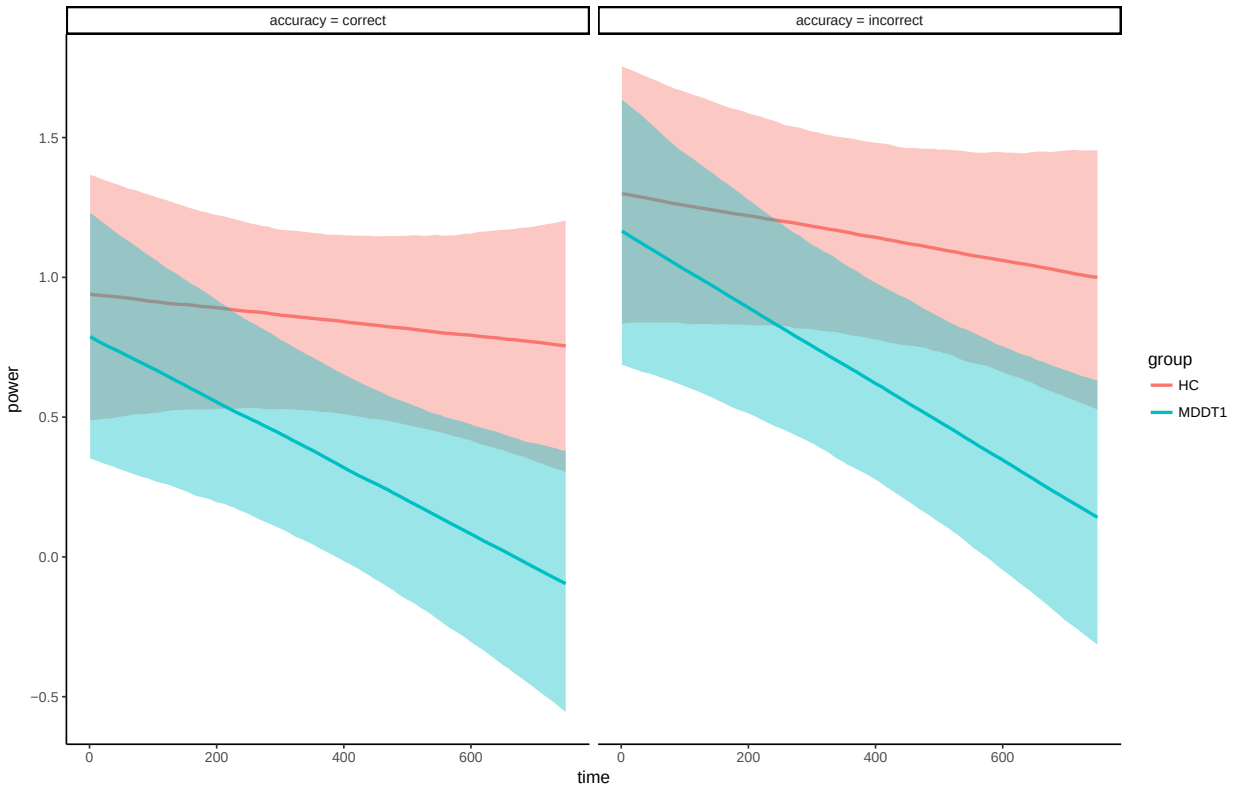

cor_snG__Intercept__time
cor_snG__Intercept__time
cor_snG__Intercept__accuracyincorrect
_snG__Intercept__accuracyincorı
cor_snG__time__accuracyincorrect
or_snG__time__accuracyincorrec
cor_snG__Intercept__time:accuracyincorrect
ıG__Intercept__time:accuracyinc
cor_snG__time__time:accuracyincorrect
_snG__time__time:accuracyincor
−0.5
0.0
0.5
0
200
400
600
800
1000
Chain
1
2
3
4



Vizualize predictions:

```
# playing around to tune ggplot options
meconditions <- make_conditions(ABN50, c("accuracy"))                    # condition on accuracy
me = marginal_effects(fit_timeaccuracytimegroup,
                      effects = "time:group",                            # select only time:group
                      conditions = meconditions,
                      re_formula = NA)                                   # include random effects
plotme = plot(me, ncol = 2, points = FALSE, ask = FALSE, plot = FALSE    # save output of the plo
              )
theme_set(theme_classic())
plotme
```

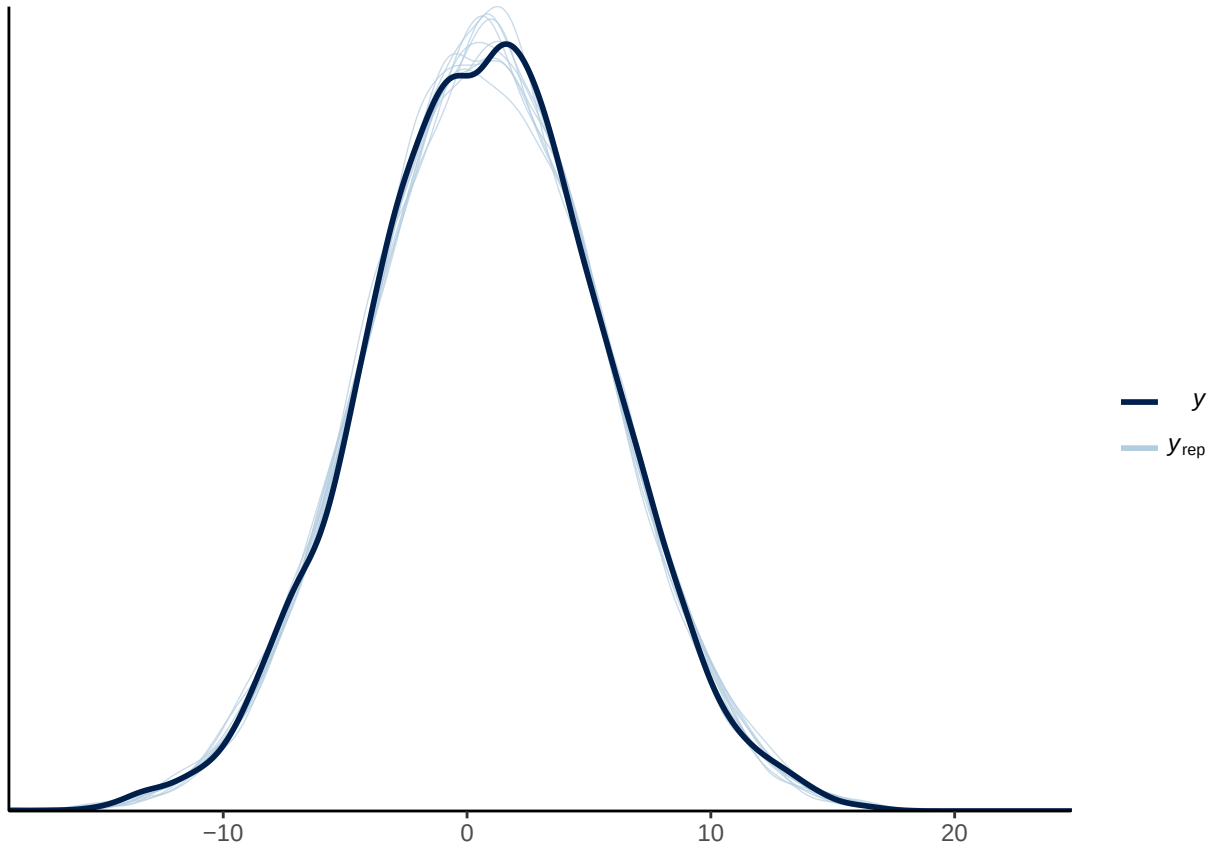
```
$`time:group`
```



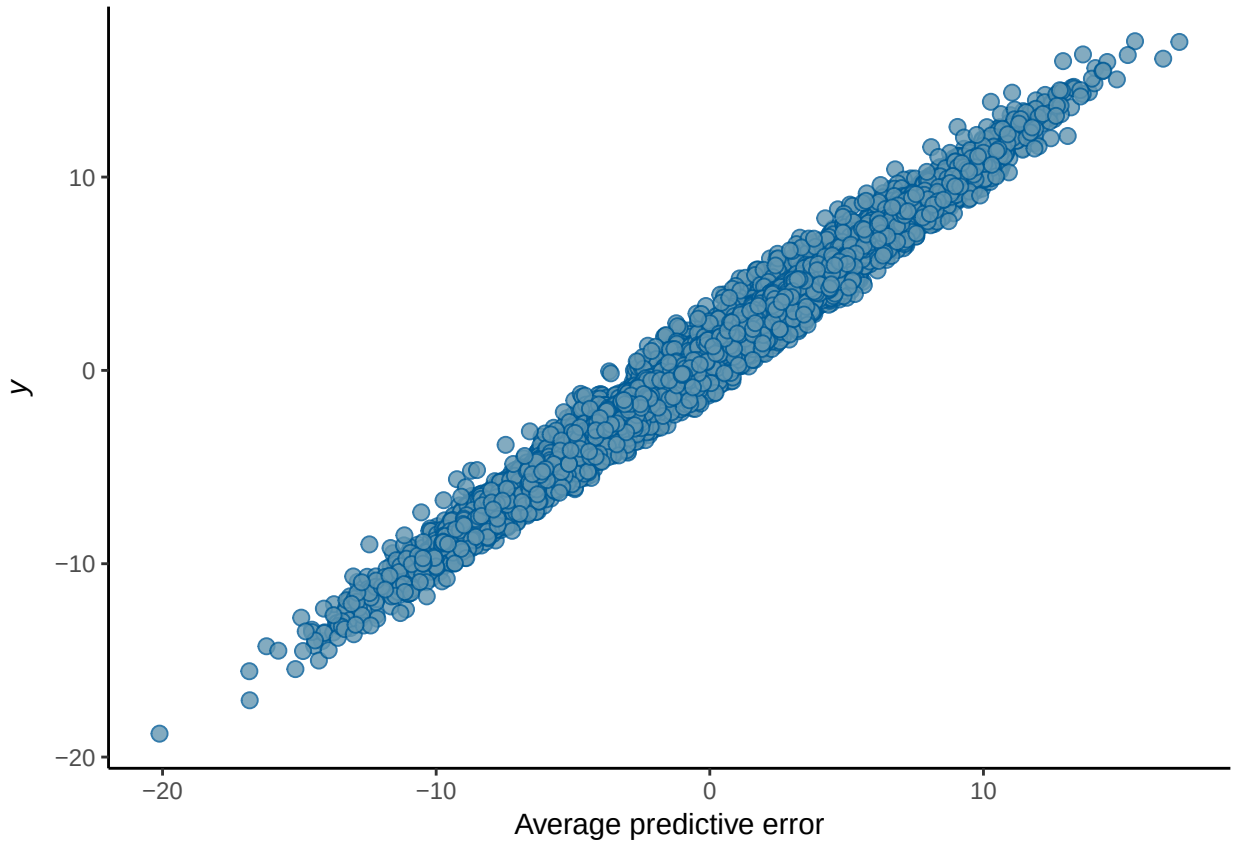
Compute fitted values and residuals

Check model fit:

```
pp_check(fit_timeaccuracytimegroup)
```




```
pp_check(fit_timeaccuracytimegroup, "error_scatter_avg")
```



6) Fit an extended model - time * accuracy * group

(main effects + interactions of time, accuracy and group - varying intercept, main effects + interactions)

```
fit_timeaccuracygroup = readRDS(file = "ABN50_singletrial_timeline_accuracy_FITtimeaccuracygroup")

#Ladislas: between design: group should not be set as varying effect
# fit_timeaccuracygroup <- brm(power ~ 1 + time*accuracy*group + (1 + time*accuracy | snG), data = ABN5
                             # cores = 8, control = list(adapt_delta = 0.99))


# saveRDS(fit_timeaccuracygroup,file = "ABN50_singletrial_timeline_accuracy_FITtimeaccuracygroup")
```

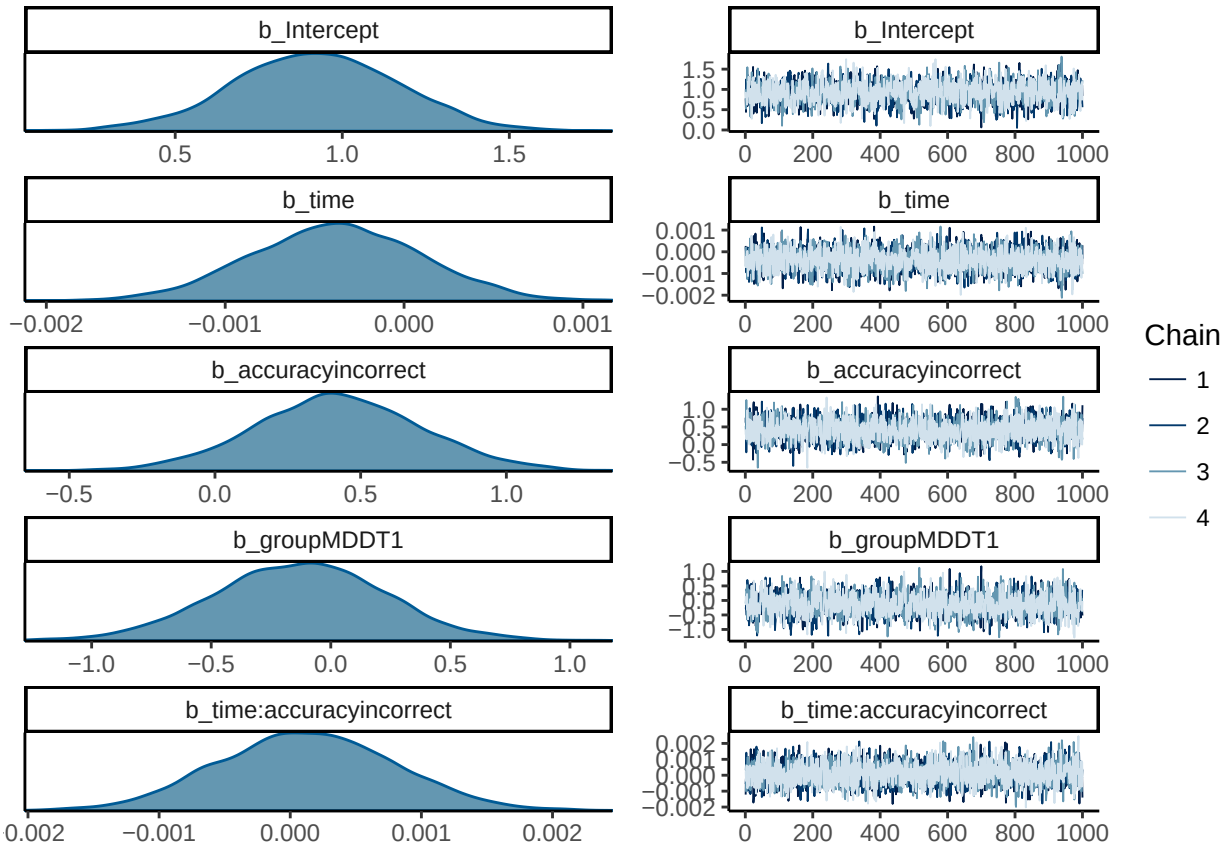
Summarize results:

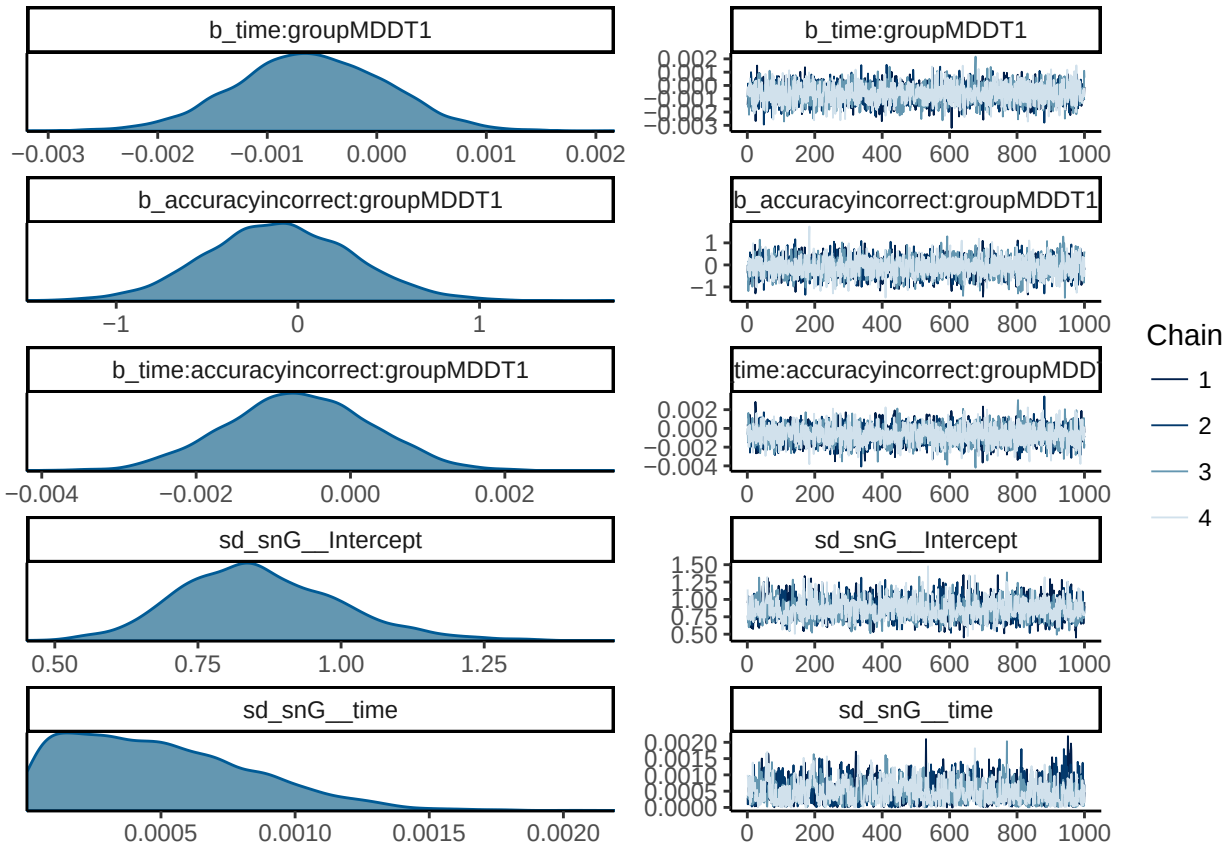
```
summary(fit_timeaccuracygroup)
```

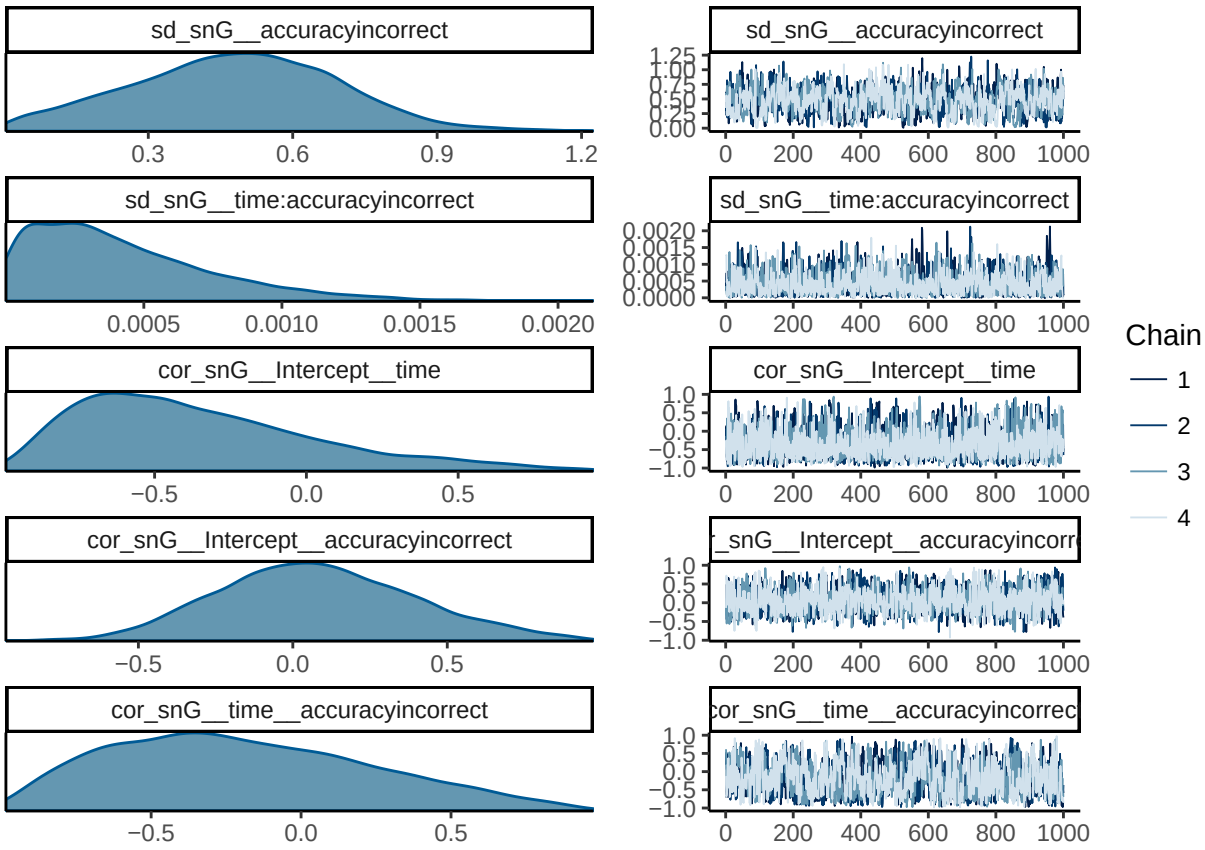
```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: power ~ 1 + time * accuracy * group + (1 + time * accuracy | snG)
   Data: ABN50 (Number of observations: 8871)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Group-Level Effects:
~snG (Number of levels: 68)
                                                Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
```

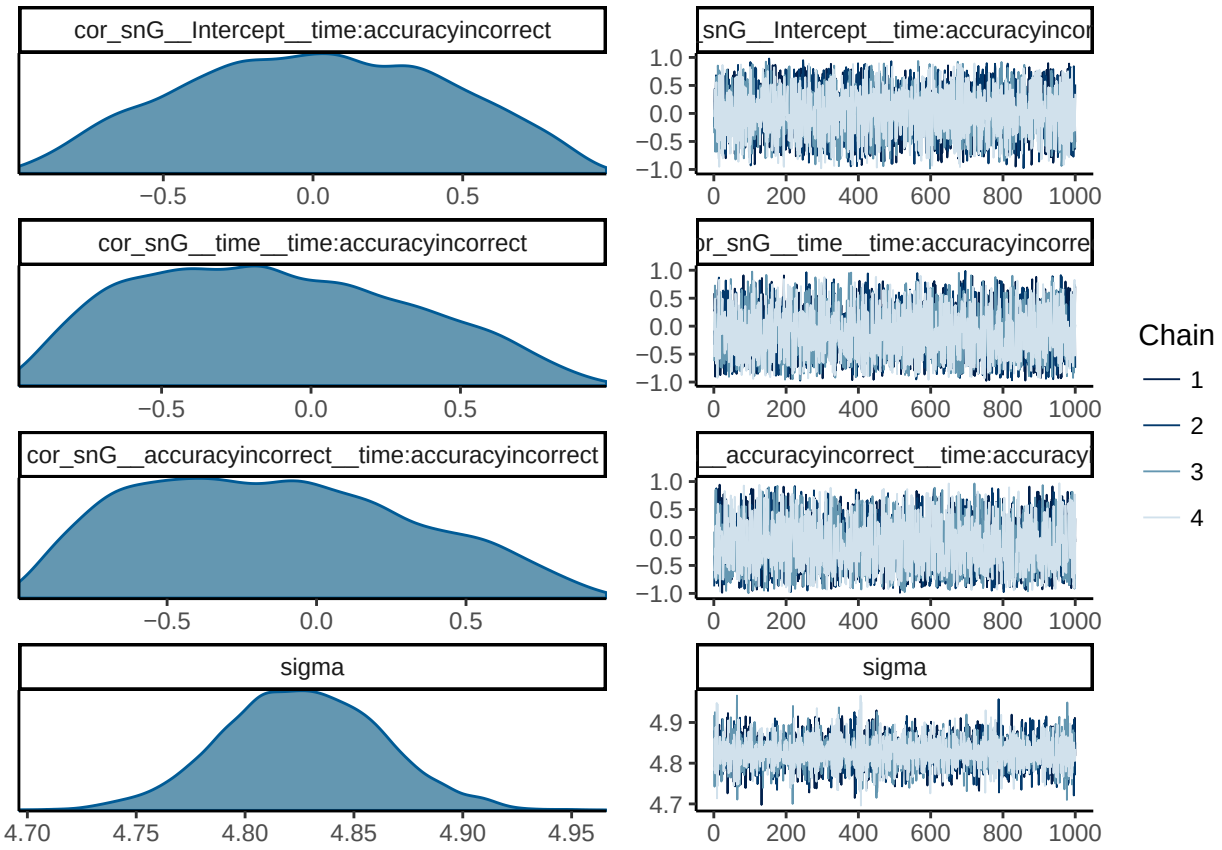
```
sd(Intercept)                                    0.86      0.14     0.61     1.16       1611 1.00
sd(time)                                         0.00      0.00     0.00     0.00        859 1.01
sd(accuracyincorrect)                            0.48      0.21     0.06     0.89        570 1.01
sd(time:accuracyincorrect)                       0.00      0.00     0.00     0.00       1401 1.00
cor(Intercept,time)                             -0.35      0.39    -0.90     0.58       2329 1.00
cor(Intercept,accuracyincorrect)                 0.08      0.31    -0.50     0.73       1546 1.00
cor(time,accuracyincorrect)                     -0.19      0.43    -0.88     0.70        689 1.00
cor(Intercept,time:accuracyincorrect)            0.01      0.43    -0.79     0.79       4000 1.00
cor(time,time:accuracyincorrect)                -0.13      0.45    -0.86     0.74       3434 1.00
cor(accuracyincorrect,time:accuracyincorrect)   -0.14      0.45    -0.88     0.74       3117 1.00

Population-Level Effects:
                                 Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept                            0.92      0.25     0.42     1.40       1864 1.00
time                                -0.00      0.00    -0.00     0.00       3547 1.00
accuracyincorrect                    0.42      0.29    -0.14     1.00       2790 1.00
groupMDDT1                          -0.13      0.36    -0.84     0.57       1619 1.00
time:accuracyincorrect               0.00      0.00    -0.00     0.00       4000 1.00
time:groupMDDT1                     -0.00      0.00    -0.00     0.00       4000 1.00
accuracyincorrect:groupMDDT1        -0.10      0.41    -0.90     0.71       2728 1.00
time:accuracyincorrect:groupMDDT1   -0.00      0.00    -0.00     0.00       3334 1.00

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma     4.83      0.04     4.75     4.90       4000 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot(fit_timeaccuracygroup, ask = FALSE)
```


b_Intercept
b_time
b_accuracyincorrect
b_groupMDDT1
b_time:accuracyincorrect
Chain
1
2
3
4

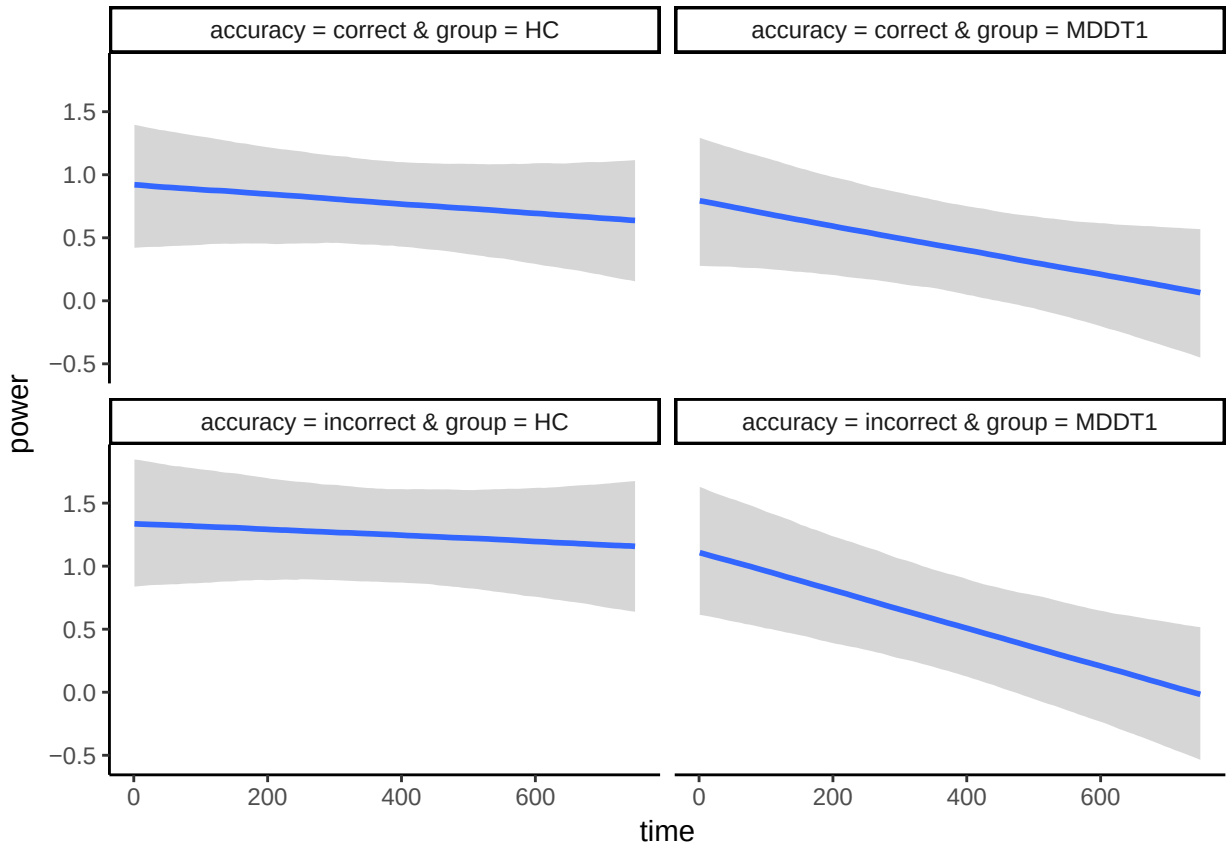

b_time:groupMDDT1
b_accuracyincorrect:groupMDDT1
b_time:accuracyincorrect:groupMDDT1
sd_snG__Intercept
sd_snG__time
time:accuracyincorrect:groupMDD
Chain
1
2
3
4

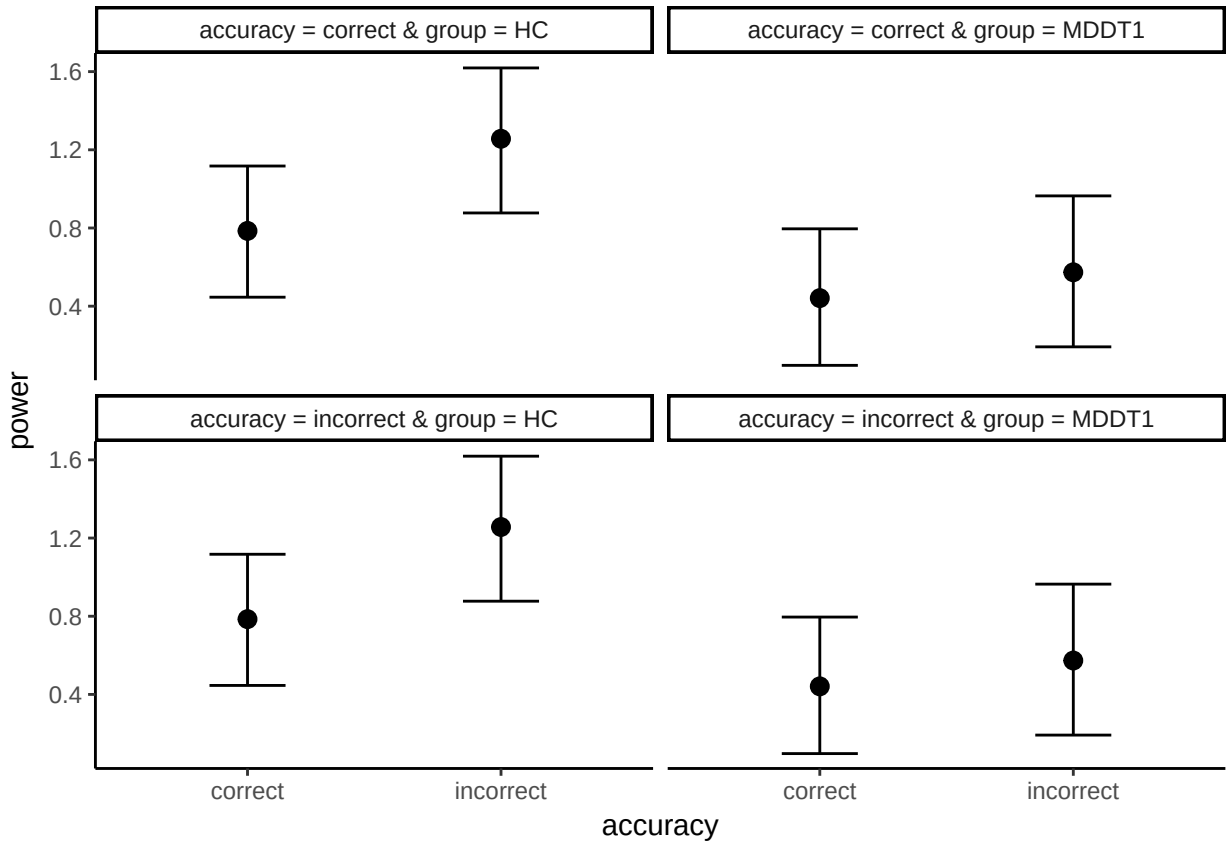

sd_snG__accuracyincorrect
0.3
0.6
0.9
1.2
sd_snG__time:accuracyincorrect
0.0005
0.0010
0.0015
0.0020
cor_snG__Intercept__time
−0.5
0.0
0.5
cor_snG__Intercept__accuracyincorrect
−0.5
0.0
0.5
cor_snG__time__accuracyincorrect
−0.5
0.0
0.5
sd_snG__accuracyincorrect
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000
sd_snG__time:accuracyincorrect
0.0020
0.0015
0.0010
0.0005
0.0000
0
200
400
600
800
1000
cor_snG__Intercept__time
1.0
0.5
0.0
−0.5
−1.0
0
200
400
600
800
1000
_snG__Intercept__accuracyincorr
1.0
0.5
0.0
−0.5
−1.0
0
200
400
600
800
1000
cor_snG__time__accuracyincorrec
1.0
0.5
0.0
−0.5
−1.0
0
200
400
600
800
1000
Chain
1
2
3
4

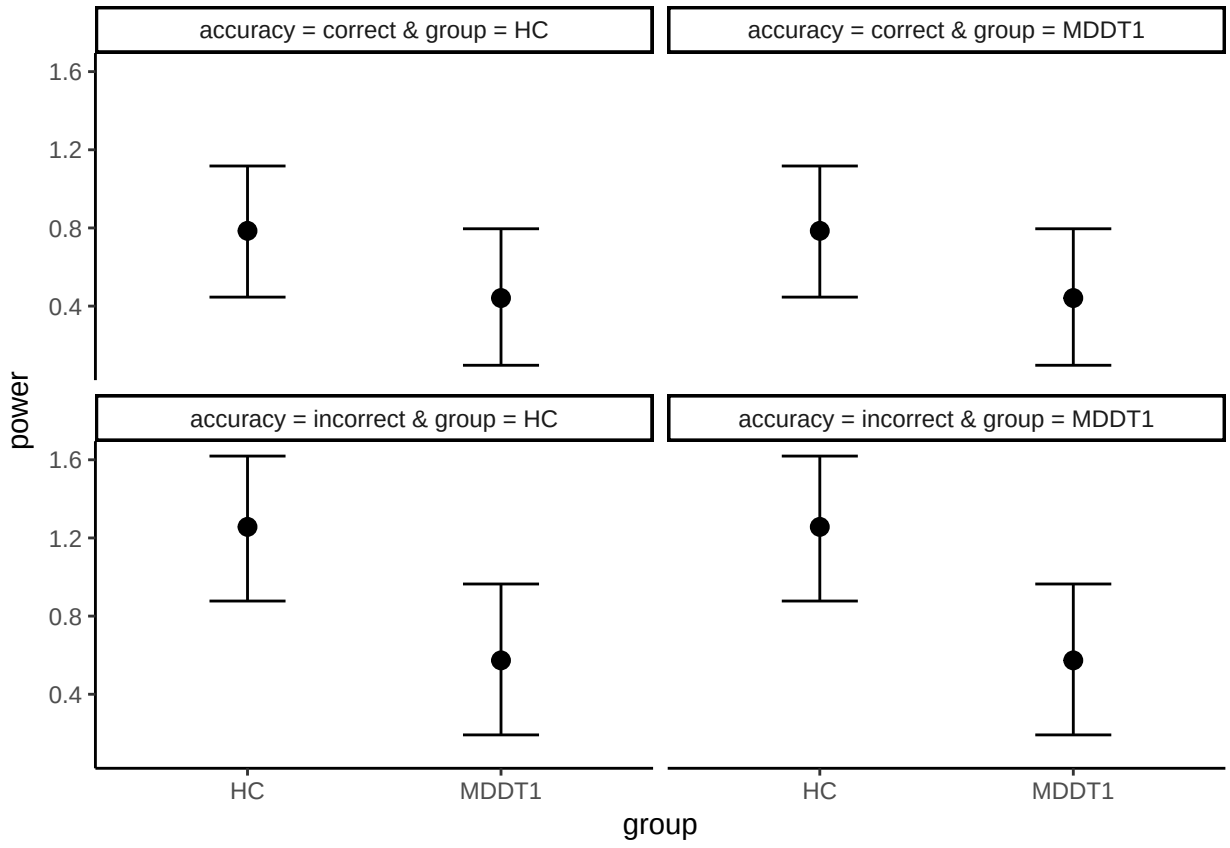


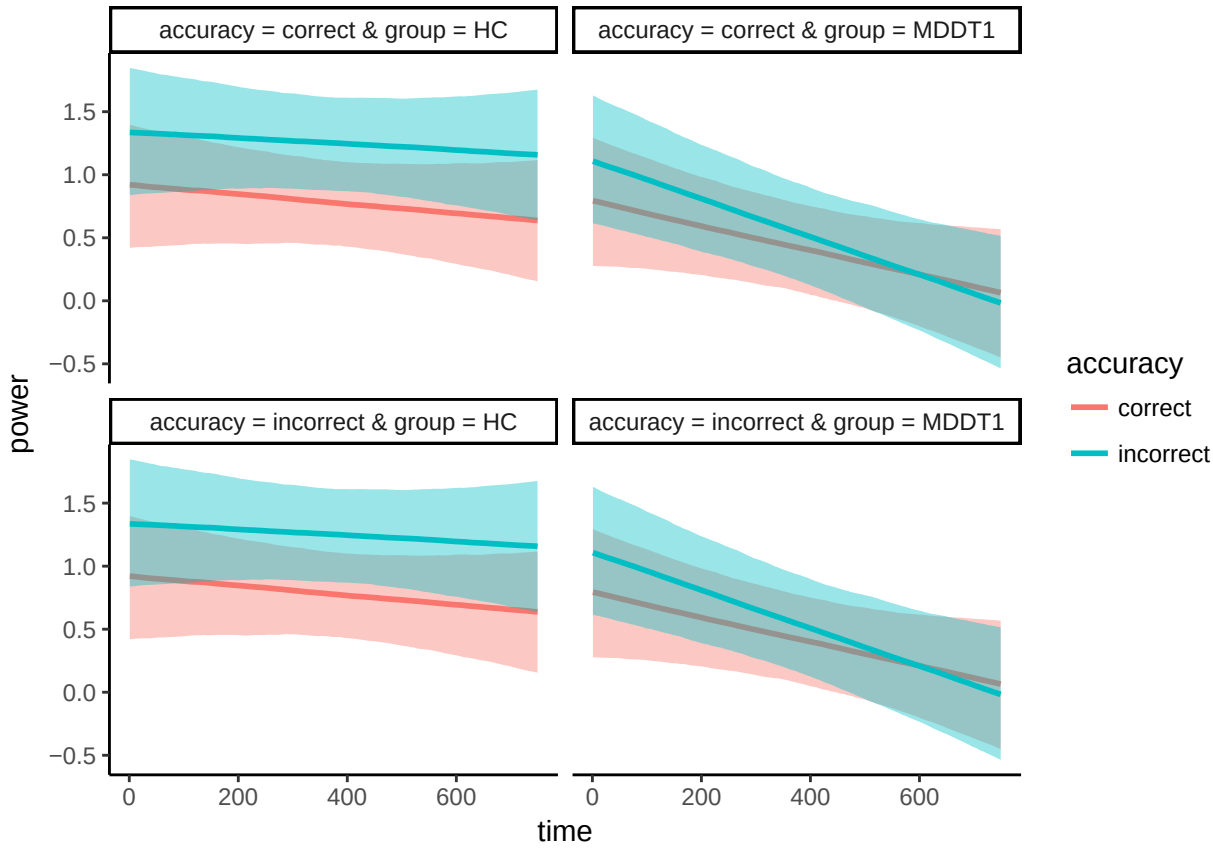
Vizualize predictions:

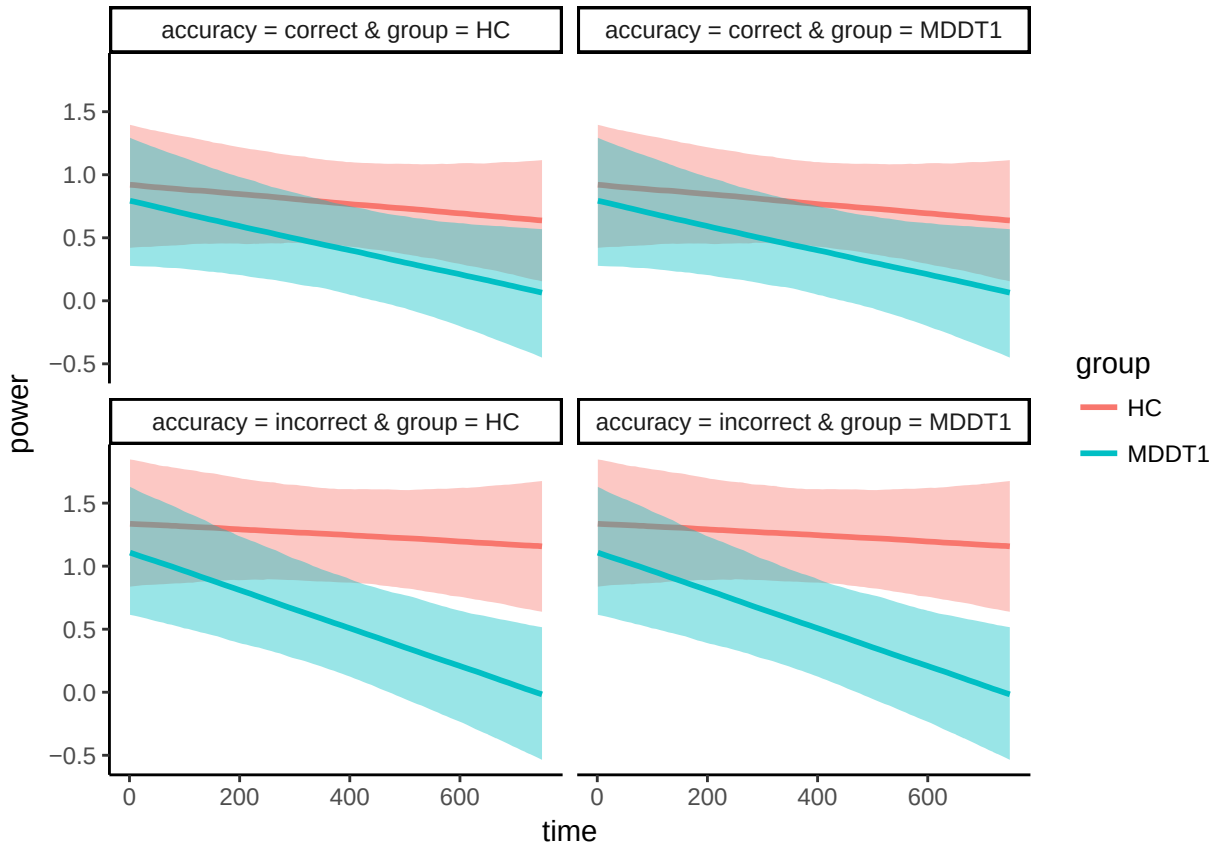
```
# re_formula = NULL ensures that group-level effects are included
me4 <- marginal_effects(fit_timeaccuracygroup, conditions = conditions,
                        re_formula = NA)
plot(me4, ncol = 2, points = FALSE, ask = FALSE)
```

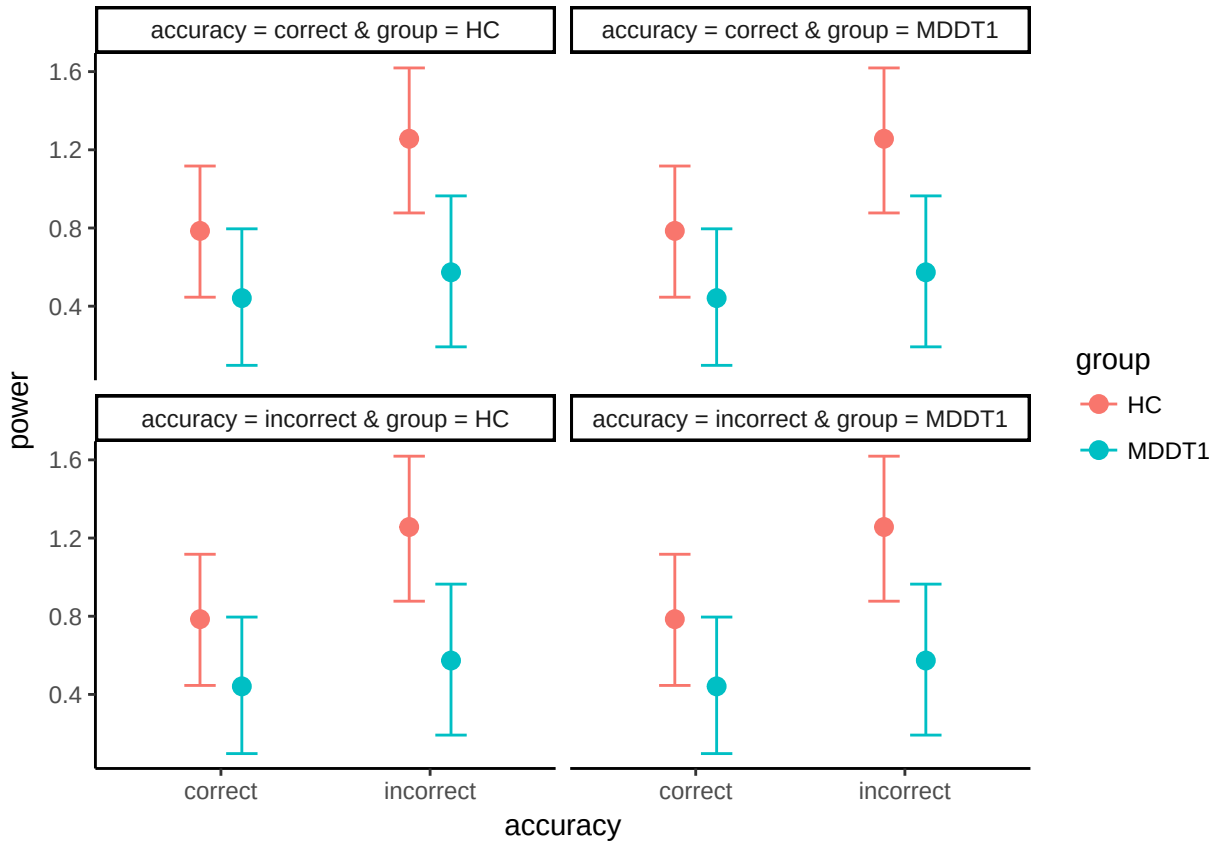

accuracy = correct & group = HC
accuracy = correct & group = MDDT1
accuracy = incorrect & group = HC
accuracy = incorrect & group = MDDT1
power
time
1.5
1.0
0.5
0.0
−0.5
0
200
400
600


accuracy = correct & group = HC
accuracy = correct & group = MDDT1
accuracy = incorrect & group = HC
accuracy = incorrect & group = MDDT1
power
1.6
1.2
0.8
0.4
correct
incorrect
accuracy


accuracy = correct & group = HC
accuracy = correct & group = MDDT1
accuracy = incorrect & group = HC
accuracy = incorrect & group = MDDT1
power
1.6
1.2
0.8
0.4
HC
MDDT1
group

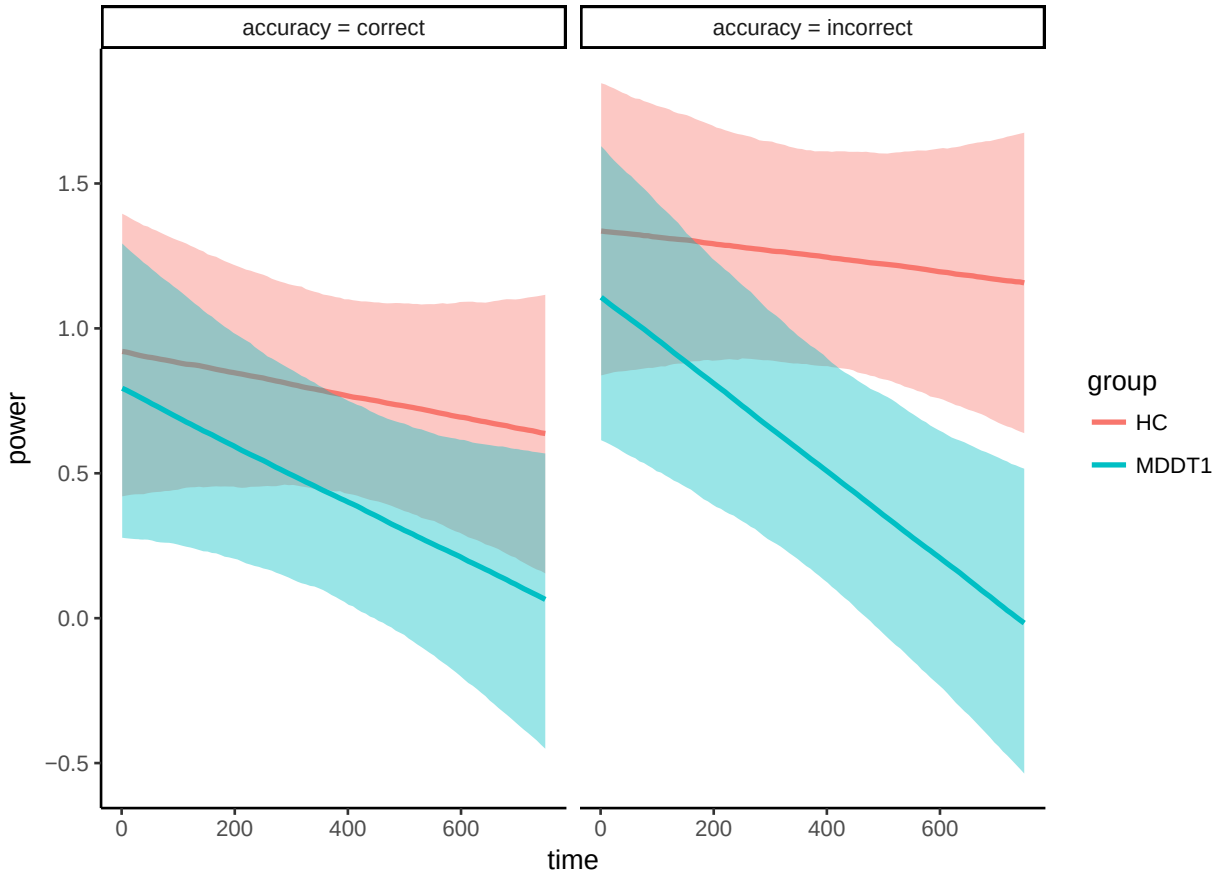

accuracy = correct & group = HC
accuracy = correct & group = MDDT1
accuracy = incorrect & group = HC
accuracy = incorrect & group = MDDT1
power
time
1.5
1.0
0.5
0.0
−0.5
0
200
400
600
accuracy
correct
incorrect


accuracy = correct & group = HC
accuracy = correct & group = MDDT1
accuracy = incorrect & group = HC
accuracy = incorrect & group = MDDT1
power
1.5
1.0
0.5
0.0
−0.5
0
200
400
600
time
group
HC
MDDT1




```
# playing around to tune ggplot options
meconditions <- make_conditions(ABN50, c("accuracy"))                    # condition on accuracy
me = marginal_effects(fit_timeaccuracygroup,
                      effects = "time:group",                            # select only time:group
                      conditions = meconditions,
                      re_formula = NA)                                   # include random effects
plotme = plot(me, ncol = 2, points = FALSE, ask = FALSE, plot = FALSE    # save output of the plo
              )
theme_set(theme_classic())
plotme
```

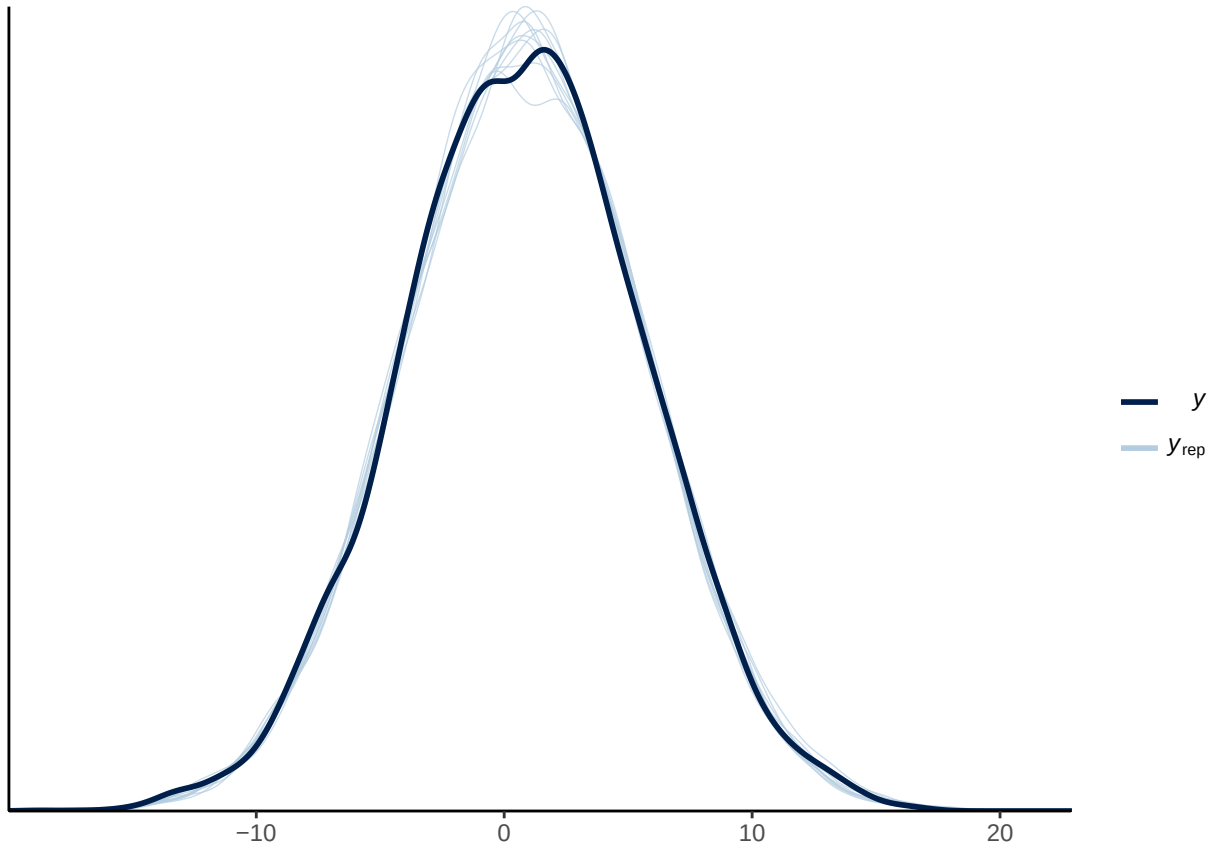
```
$`time:group`
```



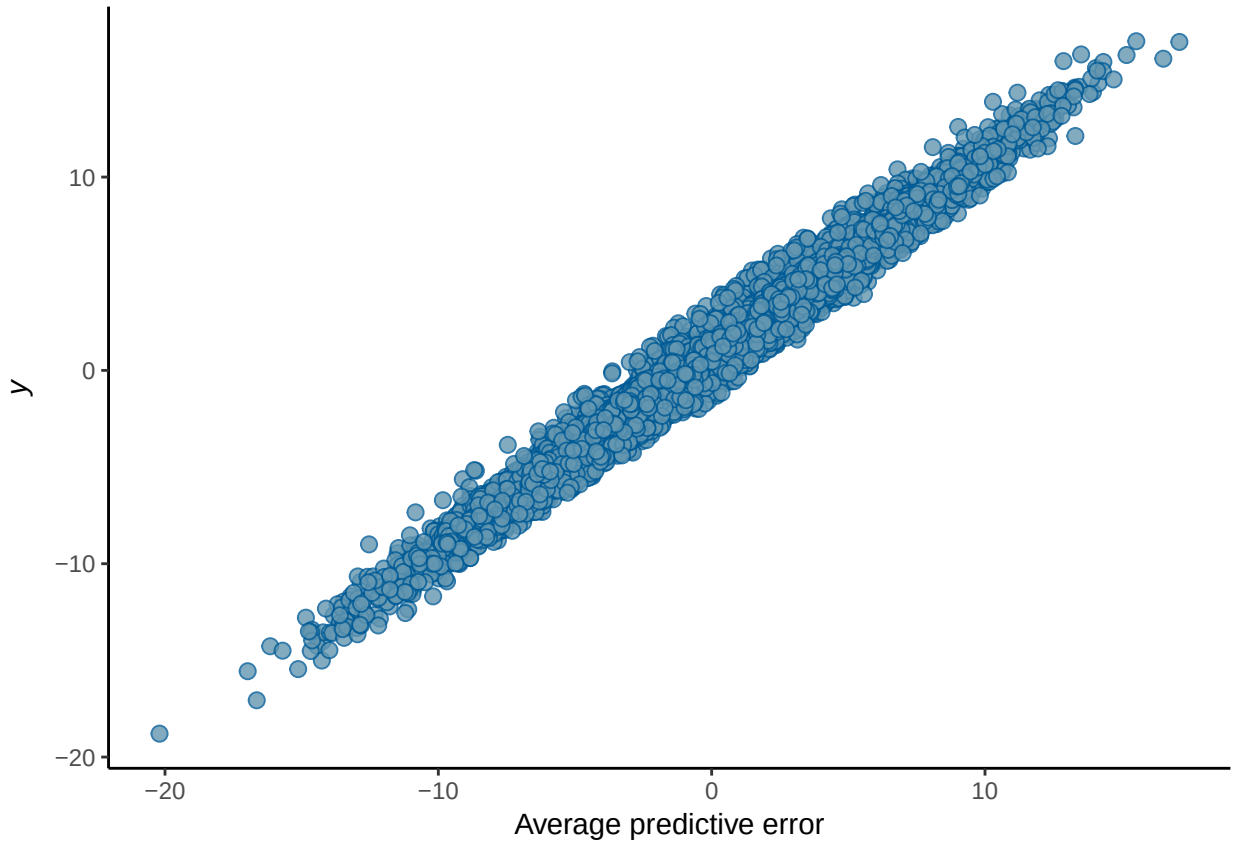
Compute fitted values and residuals

Check model fit:

```
pp_check(fit_timeaccuracygroup)
```




```
pp_check(fit_timeaccuracygroup, "error_scatter_avg")
```



Compare the 6 models

leave-one-out cross-validation, widely applicable inforation criteria

```
loo(fit_timeaccuracygroup, fit_timeaccuracytimegroup, fit_timegroup, fit_timeaccuracy, fit_compact, fit_
```

```
                                                 LOOIC     SE
fit_timeaccuracygroup                         53185.14 132.97
fit_timeaccuracytimegroup                     53183.77 132.98
fit_timegroup                                 53191.30 133.22
fit_timeaccuracy                              53186.45 132.95
fit_compact                                   53194.53 133.24
fit_null                                      53203.32 133.39
fit_timeaccuracygroup - fit_timeaccuracytimegroup     1.37   2.82
fit_timeaccuracygroup - fit_timegroup                -6.16   8.53
fit_timeaccuracygroup - fit_timeaccuracy             -1.31   4.94
fit_timeaccuracygroup - fit_compact                  -9.39   9.65
fit_timeaccuracygroup - fit_null                    -18.18  11.87
fit_timeaccuracytimegroup - fit_timegroup            -7.53   8.09
fit_timeaccuracytimegroup - fit_timeaccuracy         -2.68   4.39
fit_timeaccuracytimegroup - fit_compact             -10.76   9.26
fit_timeaccuracytimegroup - fit_null                -19.55  11.63
fit_timegroup - fit_timeaccuracy                      4.85   9.37
fit_timegroup - fit_compact                          -3.23   4.61
fit_timegroup - fit_null                            -12.02   8.36
```

```
fit_timeaccuracy - fit_compact                       -8.08   8.28
fit_timeaccuracy - fit_null                         -16.87  10.84
fit_compact - fit_null                               -8.79   6.93
```

```
WAIC(fit_timeaccuracygroup, fit_timeaccuracytimegroup, fit_timegroup, fit_timeaccuracy, fit_compact, fi
```

```
                                                    WAIC     SE
fit_timeaccuracygroup                           53184.81 132.96
fit_timeaccuracytimegroup                       53183.44 132.97
fit_timegroup                                   53191.09 133.22
fit_timeaccuracy                                53186.12 132.95
fit_compact                                     53194.32 133.24
fit_null                                        53203.15 133.39
fit_timeaccuracygroup - fit_timeaccuracytimegroup     1.37   2.82
fit_timeaccuracygroup - fit_timegroup                -6.28   8.53
fit_timeaccuracygroup - fit_timeaccuracy             -1.32   4.94
fit_timeaccuracygroup - fit_compact                  -9.51   9.65
fit_timeaccuracygroup - fit_null                    -18.34  11.87
fit_timeaccuracytimegroup - fit_timegroup            -7.65   8.09
fit_timeaccuracytimegroup - fit_timeaccuracy         -2.69   4.39
fit_timeaccuracytimegroup - fit_compact             -10.88   9.26
fit_timeaccuracytimegroup - fit_null                -19.71  11.63
fit_timegroup - fit_timeaccuracy                      4.96   9.37
fit_timegroup - fit_compact                          -3.23   4.61
fit_timegroup - fit_null                            -12.06   8.36
fit_timeaccuracy - fit_compact                       -8.20   8.28
fit_timeaccuracy - fit_null                         -17.02  10.84
fit_compact - fit_null                               -8.83   6.93
```

Compute model weights (WAIC)

```
model_weights(fit_timeaccuracygroup,
              fit_timeaccuracytimegroup,
              fit_timegroup,
              fit_timeaccuracy,
              fit_compact,
              fit_null,
              weights = "waic")
```

```
    fit_timeaccuracygroup fit_timeaccuracytimegroup             fit_timegroup          fit_timeaccuracy
             2.813675e-01              5.583378e-01              1.217802e-02             1.456703e-01
```

Check explained variance (fit to observed data)

```
bayes_R2(fit_null)
```

```
     Estimate   Est.Error       Q2.5      Q97.5
R2 0.02873856 0.003724356 0.02182479 0.03648861
```

```
bayes_R2(fit_compact)
```

```
     Estimate   Est.Error       Q2.5      Q97.5
R2 0.03070913 0.003756101 0.02385537 0.03855839
```

```
bayes_R2(fit_timeaccuracy)
```

```
     Estimate   Est.Error       Q2.5      Q97.5
R2 0.03484321 0.004155588 0.02693151 0.04317103
```

```
bayes_R2(fit_timegroup)
```

```
     Estimate   Est.Error       Q2.5    Q97.5
R2 0.03147587 0.003865845 0.02430252 0.039515
```

```
bayes_R2(fit_timeaccuracytimegroup)
```

```
     Estimate   Est.Error       Q2.5      Q97.5
R2 0.03578971 0.004234238 0.02754336 0.04431549
```

```
bayes_R2(fit_timeaccuracygroup)
```

```
     Estimate   Est.Error       Q2.5    Q97.5
R2 0.03590574 0.004165896 0.02802145 0.044253
```

hypothesis testing

Model of interest: #6

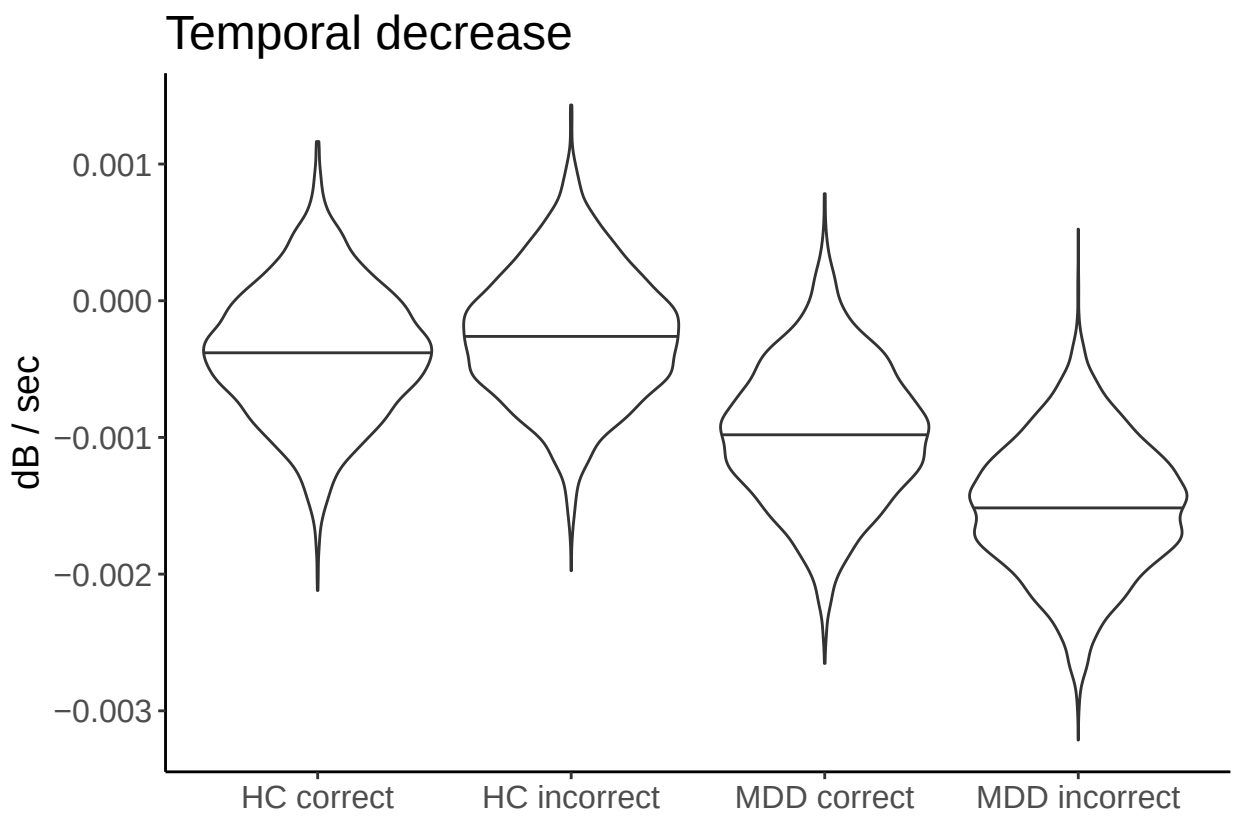
Build posterior distributions from constant effets for each condition (for: effect of time, power at time=0)

```
post <- posterior_samples(fit_timeaccuracygroup,"^b") # extracting posterior samples for constant (fixe

# build posterior distributions of each condition for SLOPE (effect of time)

slopeHCcorrect = post[["b_time"]]

slopeHCincorrect = post[["b_time"]] +
  post[["b_time:accuracyincorrect"]]

slopeMDDcorrect = post[["b_time"]] +
  post[["b_time:groupMDDT1"]]

slopeMDDincorrect = post[["b_time"]] +
  post[["b_time:groupMDDT1"]] +
  post[["b_time:accuracyincorrect"]] +
  post[["b_time:accuracyincorrect:groupMDDT1"]]



# build posterior distributions of each condition for INTERCEPT (power at time = 0)

intHCcorrect = post[["b_Intercept"]]

intHCincorrect = post[["b_Intercept"]] +
  post[["b_accuracyincorrect"]]

intMDDcorrect = post[["b_Intercept"]] +
  post[["b_groupMDDT1"]]

intMDDincorrect = post[["b_Intercept"]] +
  post[["b_groupMDDT1"]] +
```

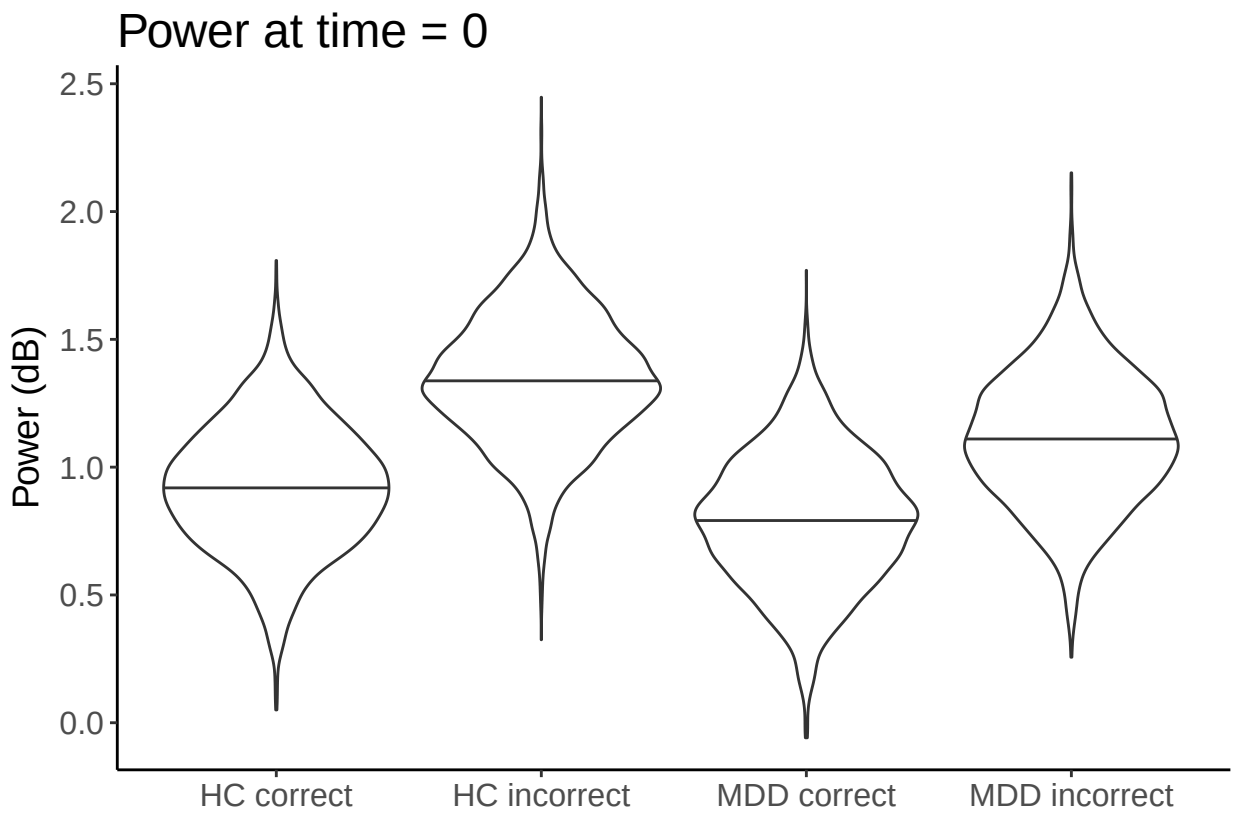
```
  post[["b_accuracyincorrect"]] +
  post[["b_accuracyincorrect:groupMDDT1"]]




# create dataframe of posterior distributions of slopes

post.power.slope = c(slopeHCcorrect, slopeHCincorrect, slopeMDDcorrect, slopeMDDincorrect)

post.label.slope = rep(c("HC correct", "HC incorrect", "MDD correct", "MDD incorrect"),  #labels
                           each = length(post.power.slope)/4)

post.slope = data.frame(post.power.slope,post.label.slope)


# create dataframe of posterior distributions of intercepts (time 0)

post.power.int = c(intHCcorrect, intHCincorrect, intMDDcorrect, intMDDincorrect)

post.label.int = rep(c("HC correct", "HC incorrect", "MDD correct", "MDD incorrect"),
                         each = length(post.power.int)/4)

post.int = data.frame(post.power.int,post.label.int)
```

plot posterior distributions and medians (to compare roughly to observed data)

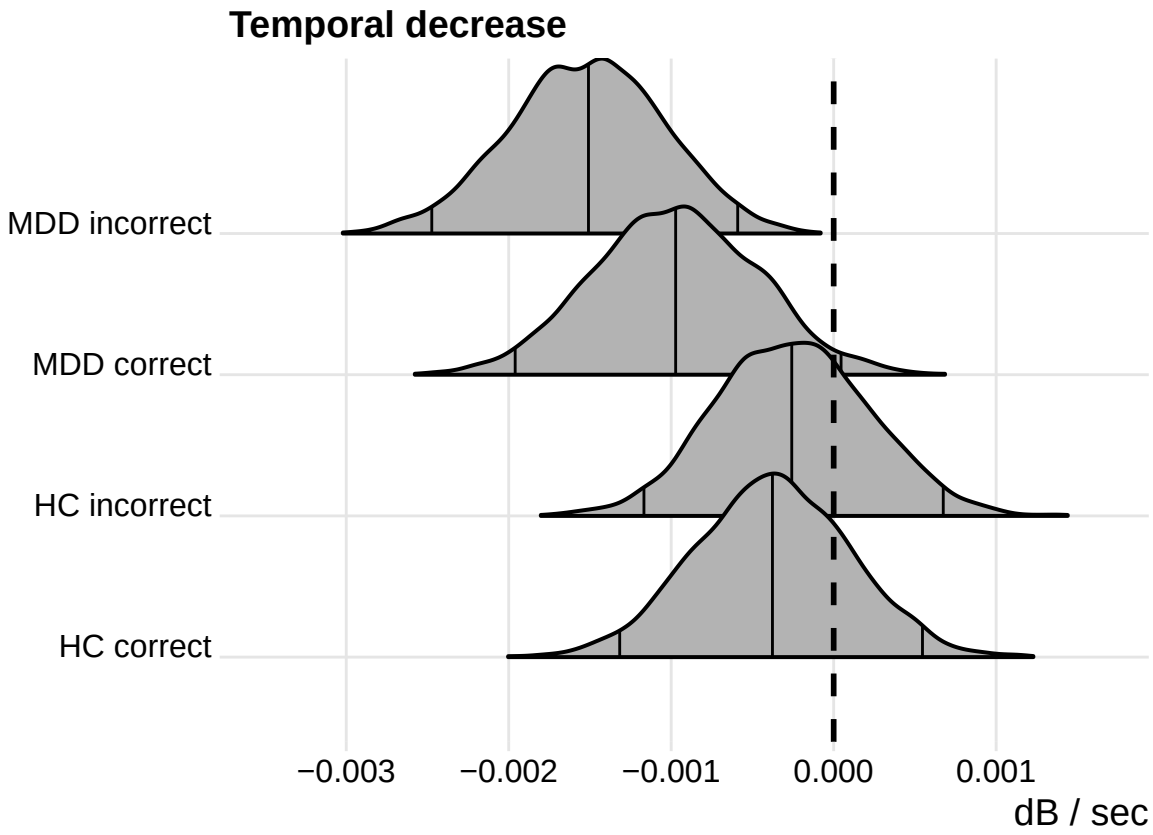
```
# violin plots

post.slope %>%
  ggplot(aes(x = post.label.slope, y = post.power.slope)) +
  geom_violin(mapping = NULL, data = NULL, stat = "ydensity",
              position = "dodge", draw_quantiles = 0.5, trim = TRUE,
              scale = "area", na.rm = FALSE, show.legend = NA, inherit.aes = TRUE) +
  # stat_summary(fun.y=mean, geom="point", shape=4, size=5, color="black") +
  theme(axis.text = element_text(size = 12),
        axis.title = element_text(size = 14),
        plot.title = element_text(size = 18)) +
  labs(title="Temporal decrease", x = "", y = "dB / sec")                    # expression(paste(beta
```



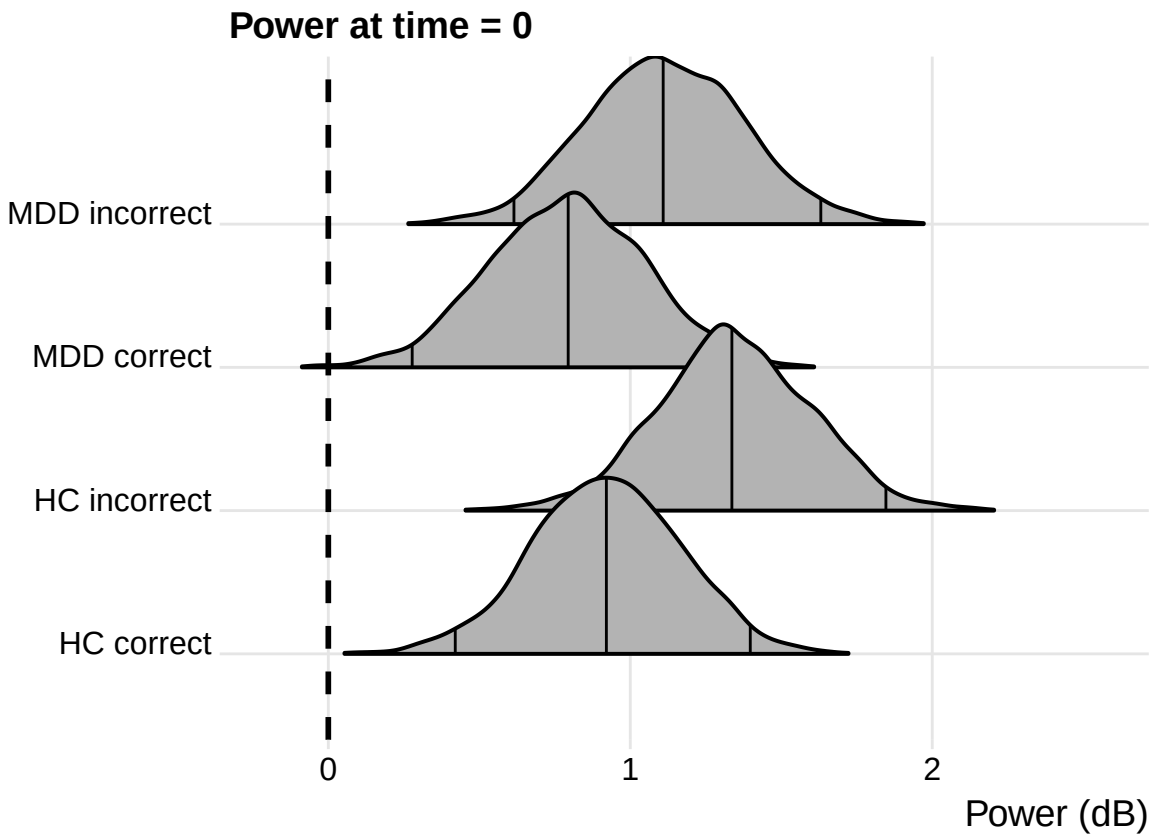

```
post.int %>%
  ggplot(aes(x = post.label.int, y = post.power.int)) +
  geom_violin(mapping = NULL, data = NULL, stat = "ydensity",
              position = "dodge", draw_quantiles = 0.5, trim = TRUE,
              scale = "area", na.rm = FALSE, show.legend = NA, inherit.aes = TRUE) +
  # stat_summary(fun.y=mean, geom="point", shape=4, size=5, color="black") +
  theme(axis.text = element_text(size = 12),
        axis.title = element_text(size = 14),
        plot.title = element_text(size = 18)) +
  labs(title="Power at time = 0", x = "", y = "Power (dB)")                    # expression(paste(bet
```




```
# density ridgeline plots



post.slope %>%
  ggplot(aes(x = post.power.slope, y = post.label.slope)) +
  geom_density_ridges2(rel_min_height = 0.005,
                       scale = 1.3,
                       quantile_lines = TRUE,
                       quantiles = c(0.025, 0.5, 0.975),
                       alpha = 1,
                       size = 0.7) +
  theme(axis.text = element_text(size = 14),
        axis.title = element_text(size = 14),
        plot.title = element_text(size = 18)) +
  labs(title="Temporal decrease", x = "dB / sec", y = "") +                    # expression(paste(be
  geom_vline(xintercept = 0, linetype = "dashed", size = 1) +
  theme_ridges()
```

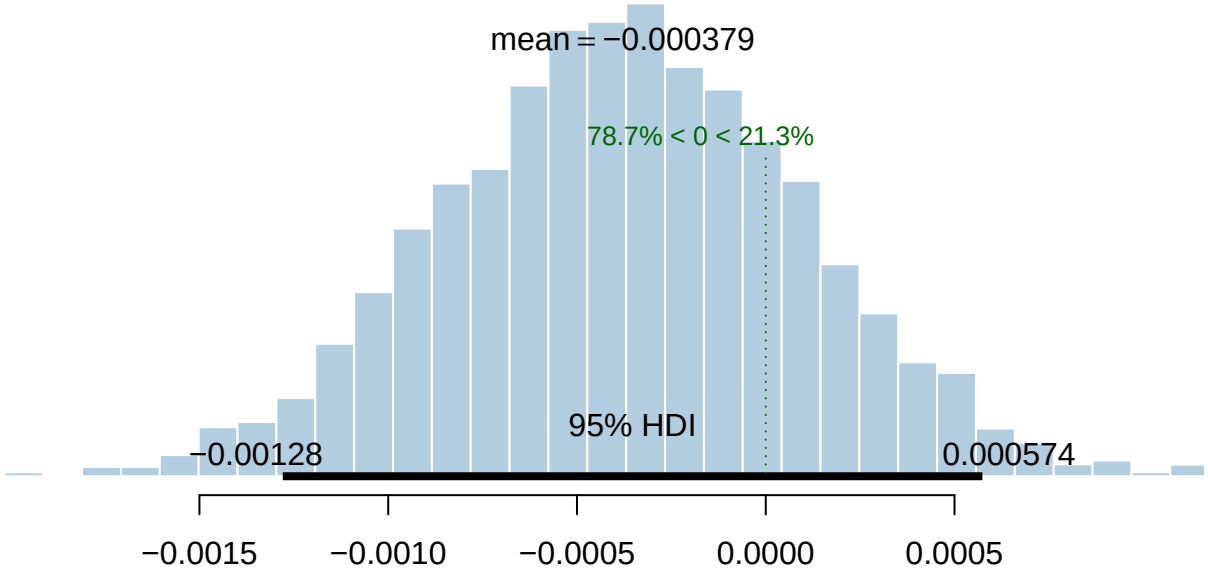



```
post.int %>%
  ggplot(aes(x = post.power.int, y = post.label.int)) +
  geom_density_ridges2(rel_min_height = 0.005,
                       scale = 1.3,
                       quantile_lines = TRUE,
                       quantiles = c(0.025, 0.5, 0.975),
                       alpha = 1,
                       size = 0.7) +
  theme(axis.text = element_text(size = 14),
        axis.title = element_text(size = 14),
        plot.title = element_text(size = 18)) +
  labs(title="Power at time = 0", x = "Power (dB)", y = "") +                    # expression(paste(b
  geom_vline(xintercept = 0, linetype = "dashed", size = 1) +
  theme_ridges()
```

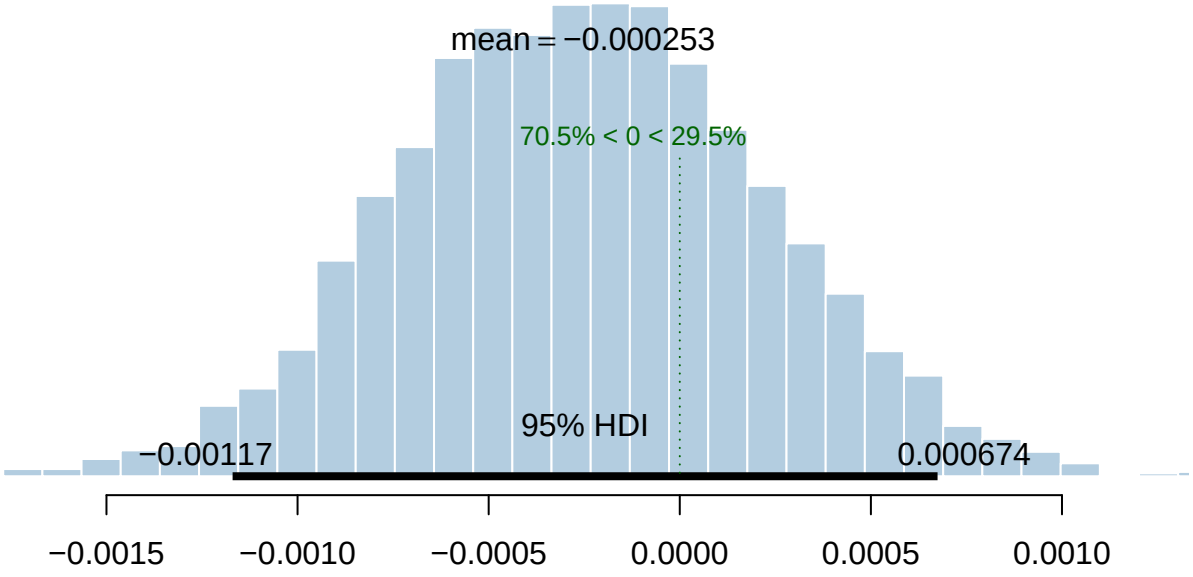


plot istograms of posterior distribution per condition (package best)

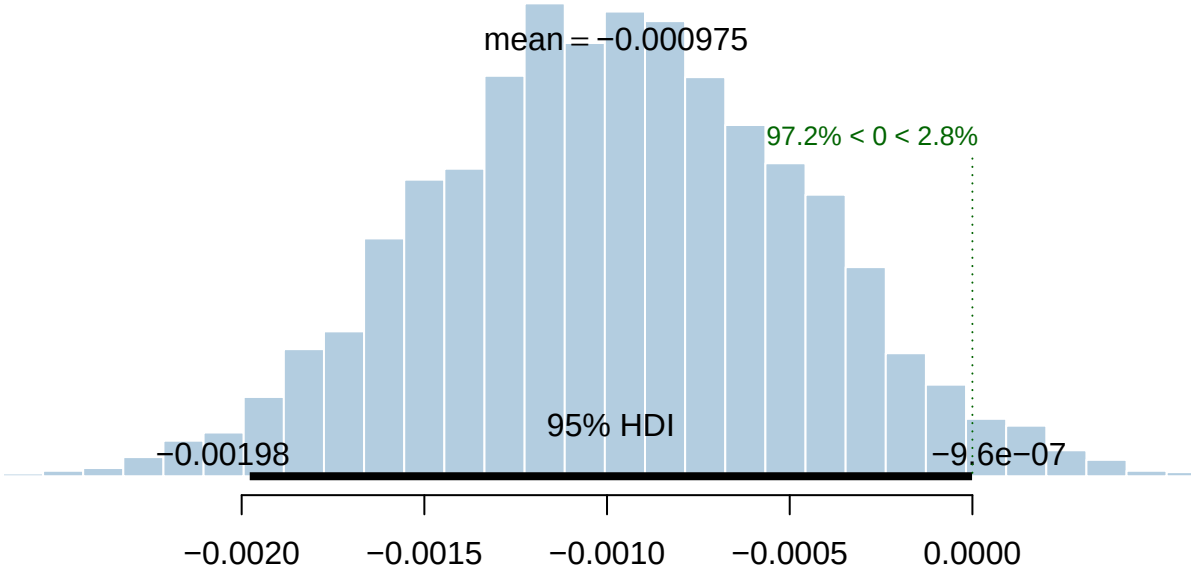
```
## slope
plotPost(slopeHCcorrect, xlab = "", col = "#b3cde0", showCurve = FALSE, cex = 1, compVal = 0)
```



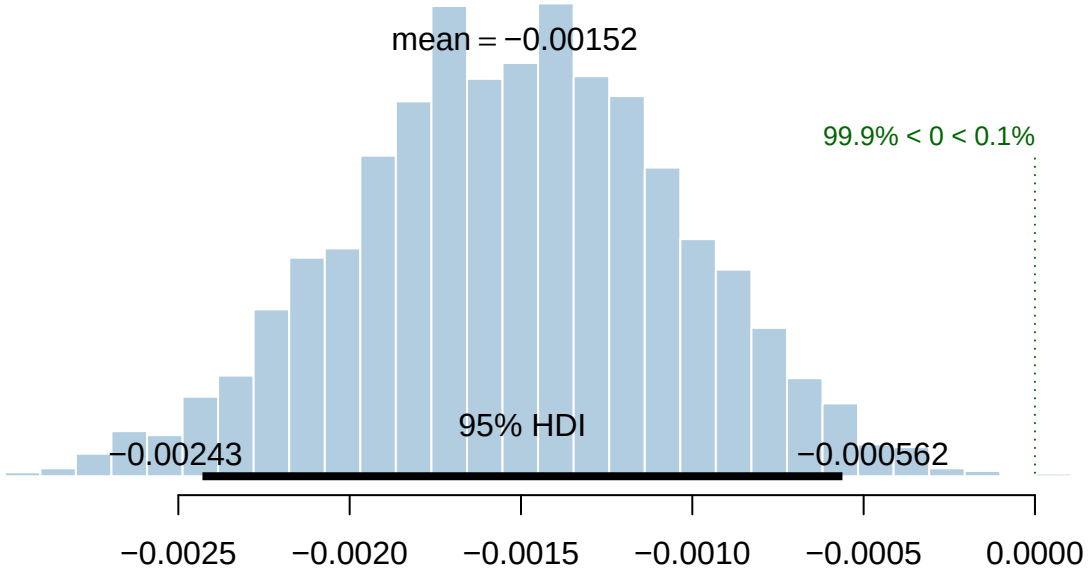

```
plotPost(slopeHCincorrect, xlab = "", col = "#b3cde0", showCurve = FALSE, cex = 1, compVal = 0)
```



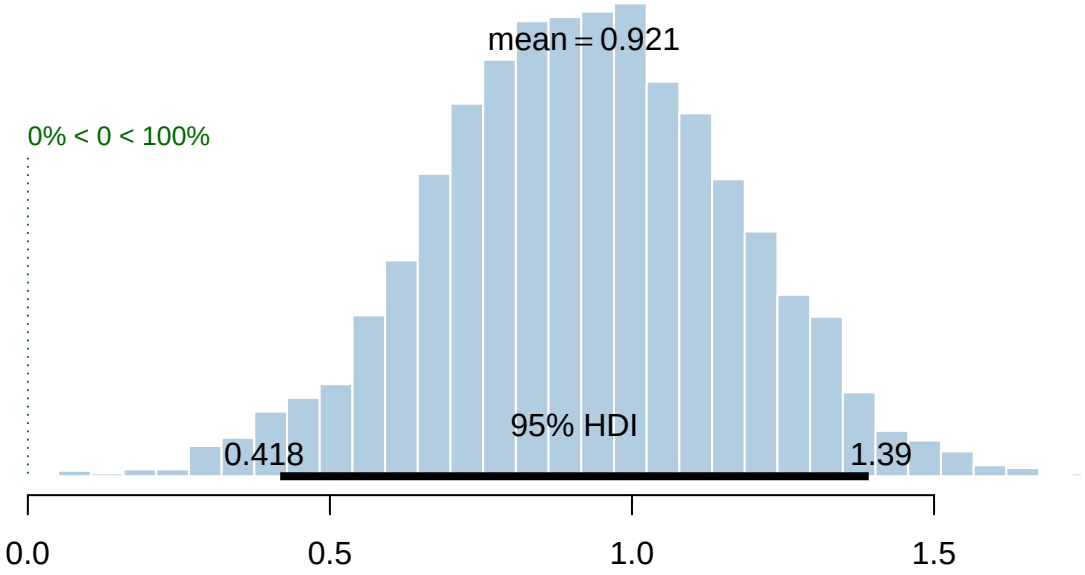

```
plotPost(slopeMDDcorrect, xlab = "", col = "#b3cde0", showCurve = FALSE, cex = 1, compVal = 0)
```



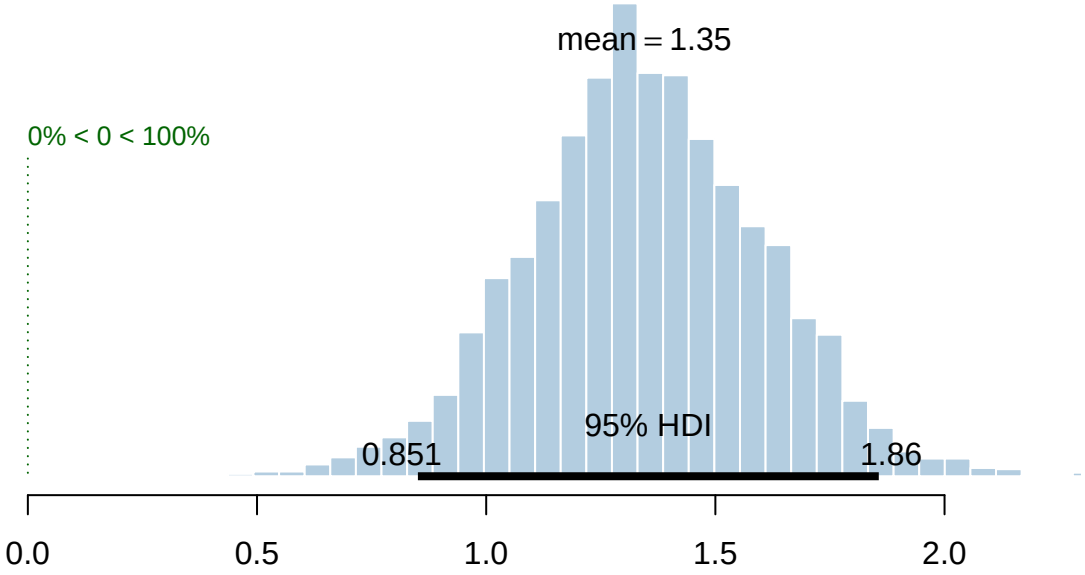

```
plotPost(slopeMDDincorrect, xlab = "", col = "#b3cde0", showCurve = FALSE, cex = 1, compVal = 0)
```



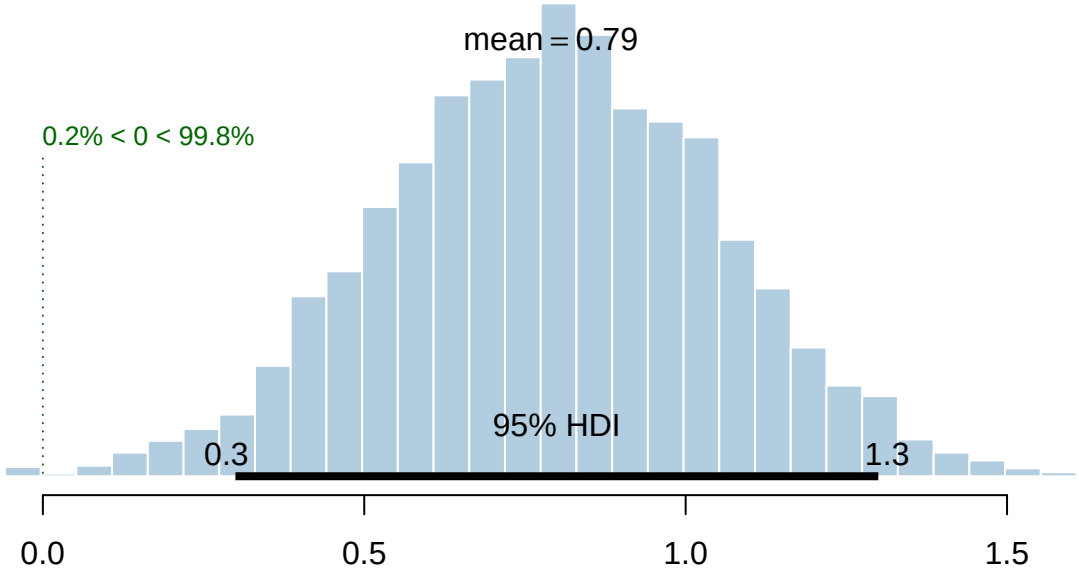

```
## intercept
plotPost(intHCcorrect, xlab = "", col = "#b3cde0", showCurve = FALSE, cex = 1, compVal = 0)
```



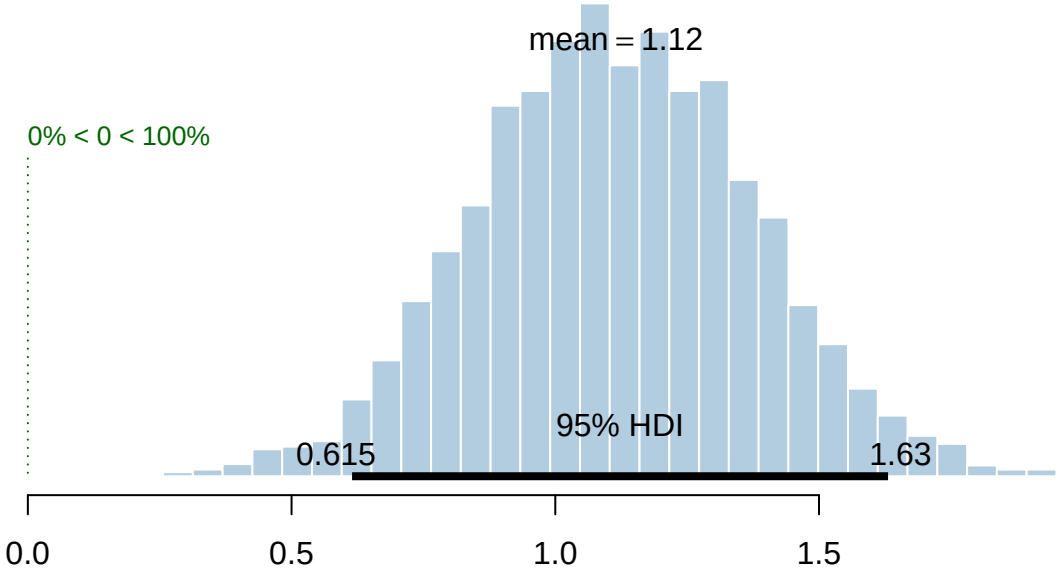

```
plotPost(intHCincorrect, xlab = "", col = "#b3cde0", showCurve = FALSE, cex = 1, compVal = 0)
```




```
plotPost(intMDDcorrect, xlab = "", col = "#b3cde0", showCurve = FALSE, cex = 1, compVal = 0)
```




```
plotPost(intMDDincorrect, xlab = "", col = "#b3cde0", showCurve = FALSE, cex = 1, compVal = 0)
```



Test some hypotheses on slope (effect of time on FMT power)

define the posterior probability distribution of slope (temporal decrease) for following contrasts:

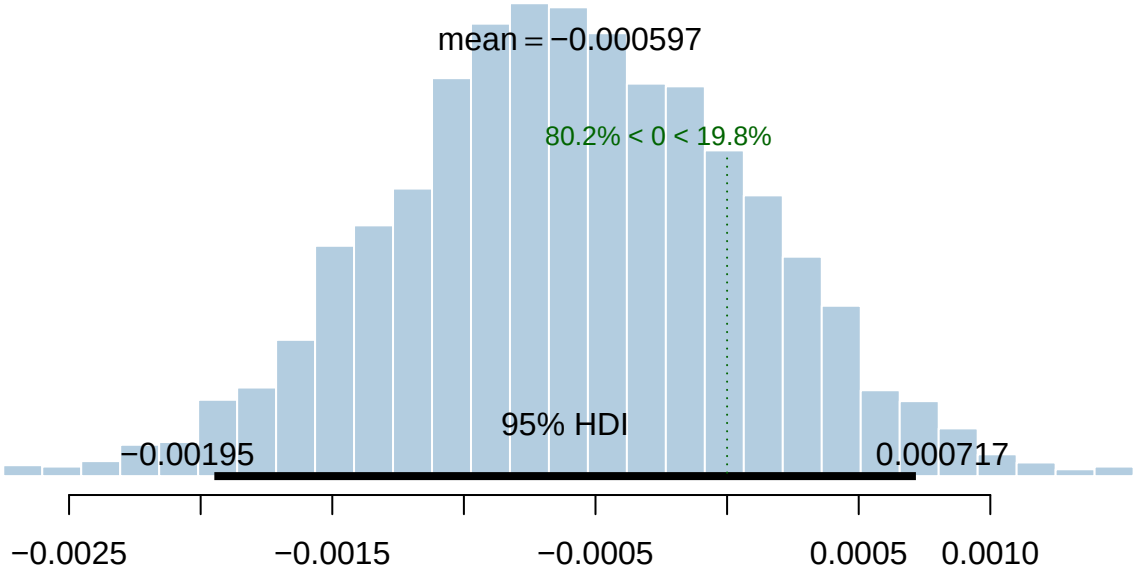
```
### slope MDD correct vs HC correct

slopecorrectDIFF = slopeMDDcorrect - slopeHCcorrect

### slope MDD incorrect vs HC incorrect

slopeincorrectDIFF = slopeMDDincorrect - slopeHCincorrect


### difference of difference: MDD incorrect-correct vs HC incorrect-correct

slopedouoblediff = (slopeMDDincorrect-slopeMDDcorrect)-(slopeHCincorrect-slopeHCcorrect)

# alternative:
# slopedouoblediff = slopeincorrectDIFF - slopecorrectDIFF
```

plot posterior probability distribution (on the above contrasts)

slope: MDD correct - HC correct

EvidenceRatio: p(MDDcorrect < HCcorrect) / p(MDDcorrect > HCcorrect): 80.2/19.8 = 4.05

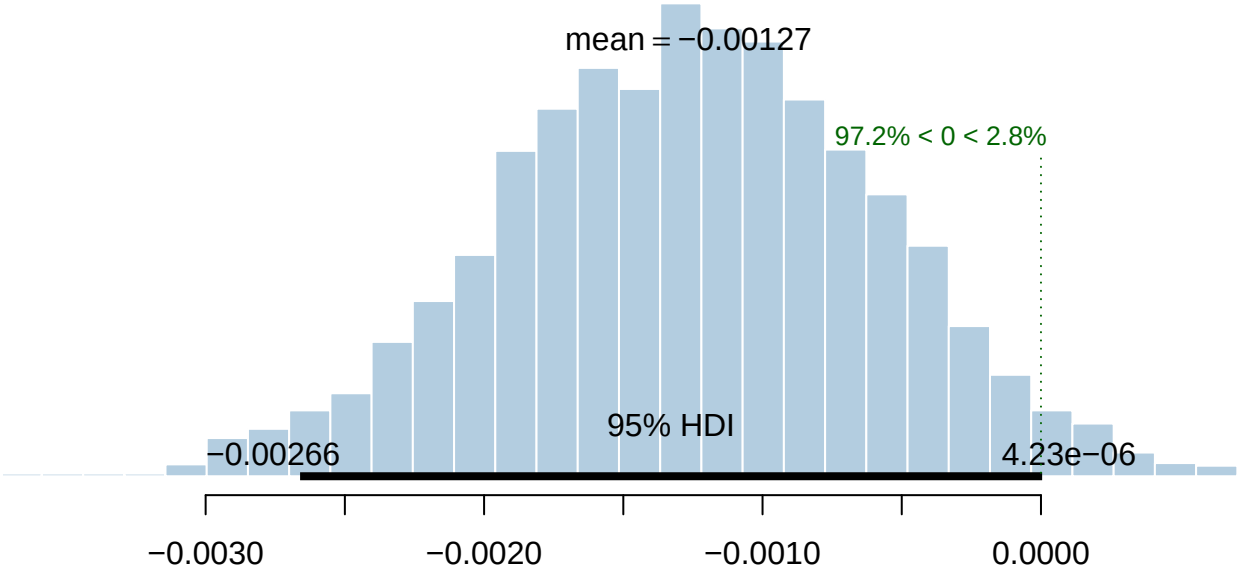
```
## slope MDD correct vs HC correct
plotPost(slopecorrectDIFF, xlab = "", col = "#b3cde0", showCurve = FALSE, cex = 1, compVal = 0)
```



slope: MDD incorrect - HC incorrect

EvidenceRatio: p(MDDincorrect < HCincorrect) / p(MDDincorrect > HCincorrect): 97.2/2.8 = 34.71

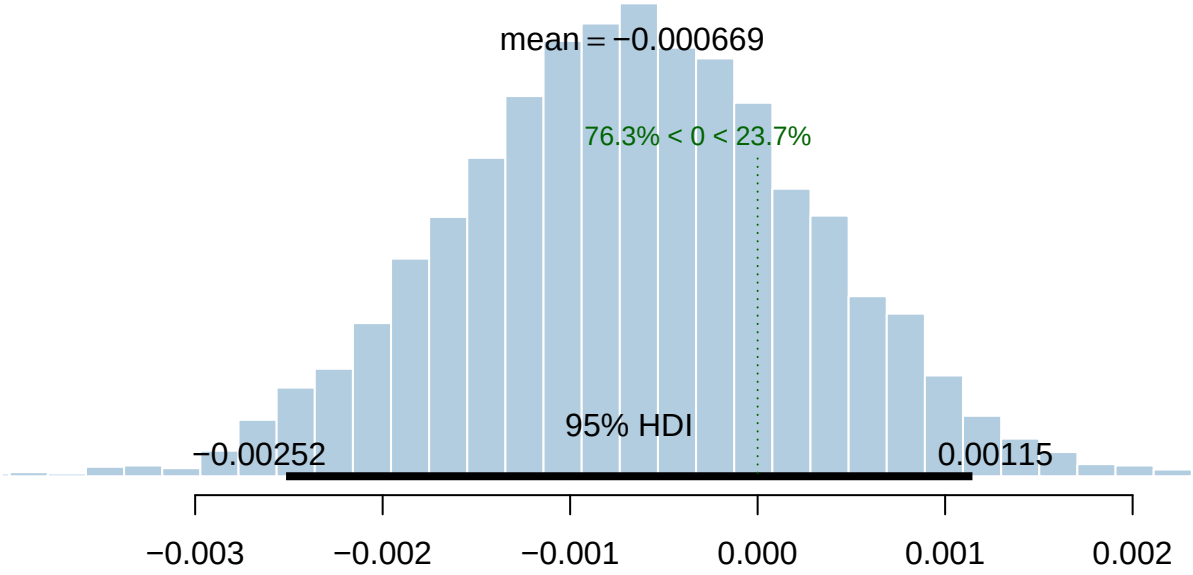
```
## slope MDD incorrect vs HC incorrect
plotPost(slopeincorrectDIFF, xlab = "", col = "#b3cde0", showCurve = FALSE, cex = 1, compVal = 0)
```



slope: MDD (incorrect-correct) - HC (incorrect-correct)

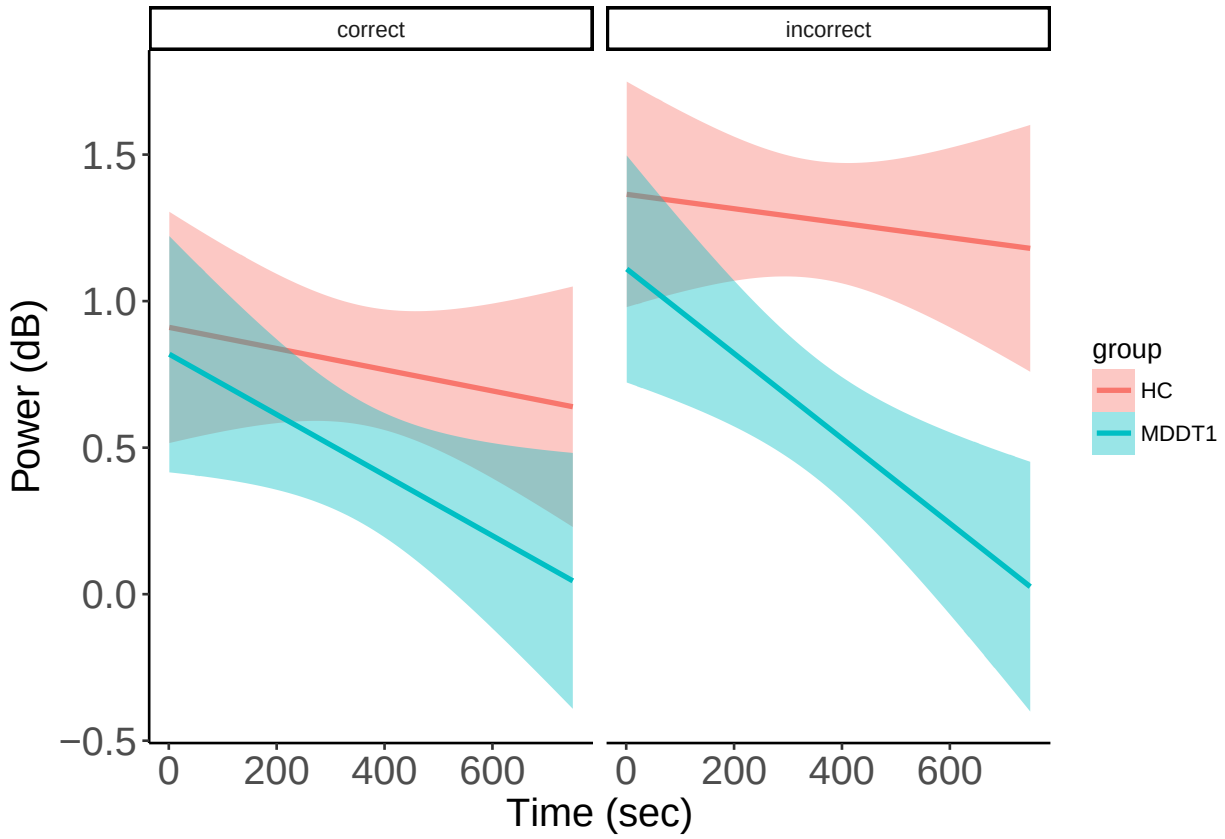
EvidenceRatio: p(MDDincorrect-correct < HCincorrect-correct) / p(MDDincorrect-correct > HCincorrect-correct): 76.3/23.7 = 3.22

```
## difference of difference: MDD(incorrectVScorrect) - HC(incorrectVScorrect)
plotPost(slopedouoblediff, xlab = "", col = "#b3cde0", showCurve = FALSE, cex = 1, compVal = 0)
```



Plot raw data (fit linear regressions)

```
# lm + 95% CI

ABN50 %>%
  ggplot(aes(x = time, y = power, color=group)) +
  # geom_point() +
  geom_smooth(method=lm, se=TRUE, fullrange=TRUE, aes(fill=group)) +
  theme(axis.text = element_text(size = 16),
        axis.title = element_text(size = 16)) +
  labs(title="", x = "Time (sec)", y = "Power (dB)") +
  facet_grid(. ~ accuracy)                                    # split according to factor
```



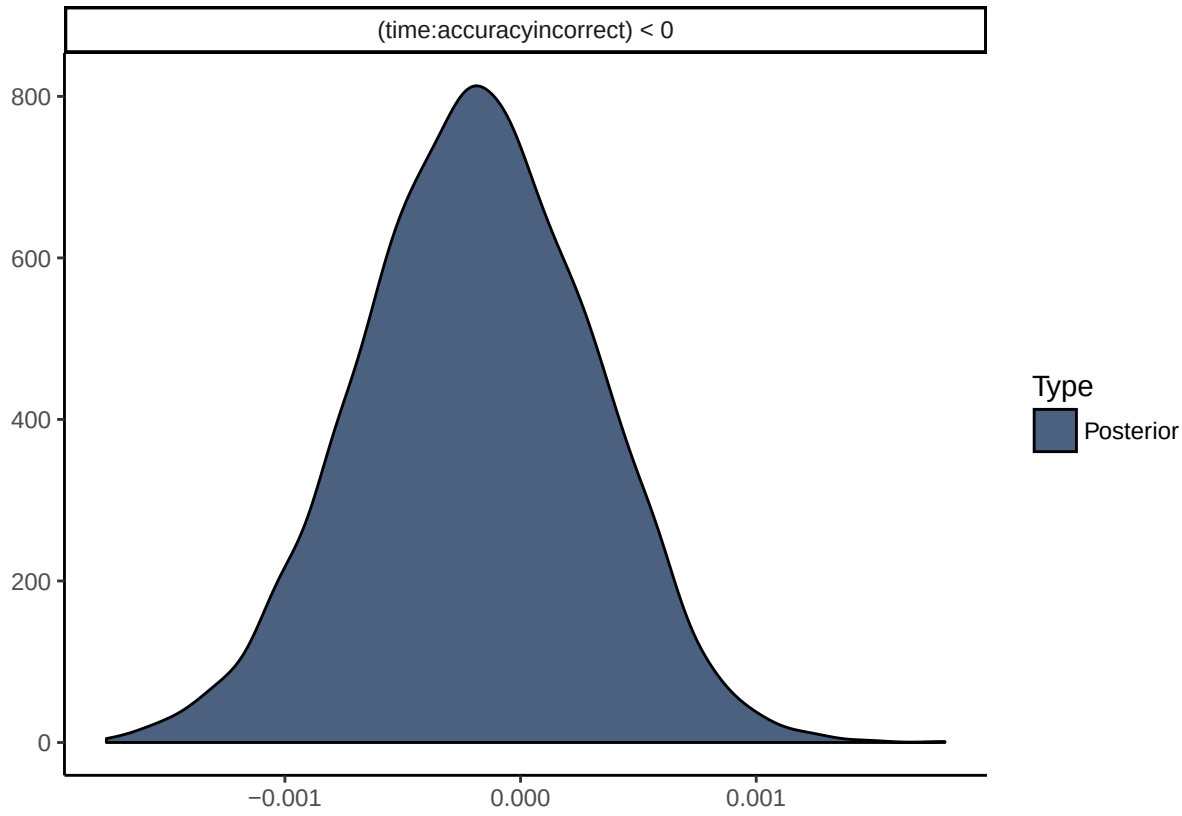
(discard - just to double check)

Evaluate some hypothesis with Hypothesis function in BRMS

Model 2: time*accuracy

1) Posterior probability that the time effect is steeper for incorrect vs correct, across the 2 groups

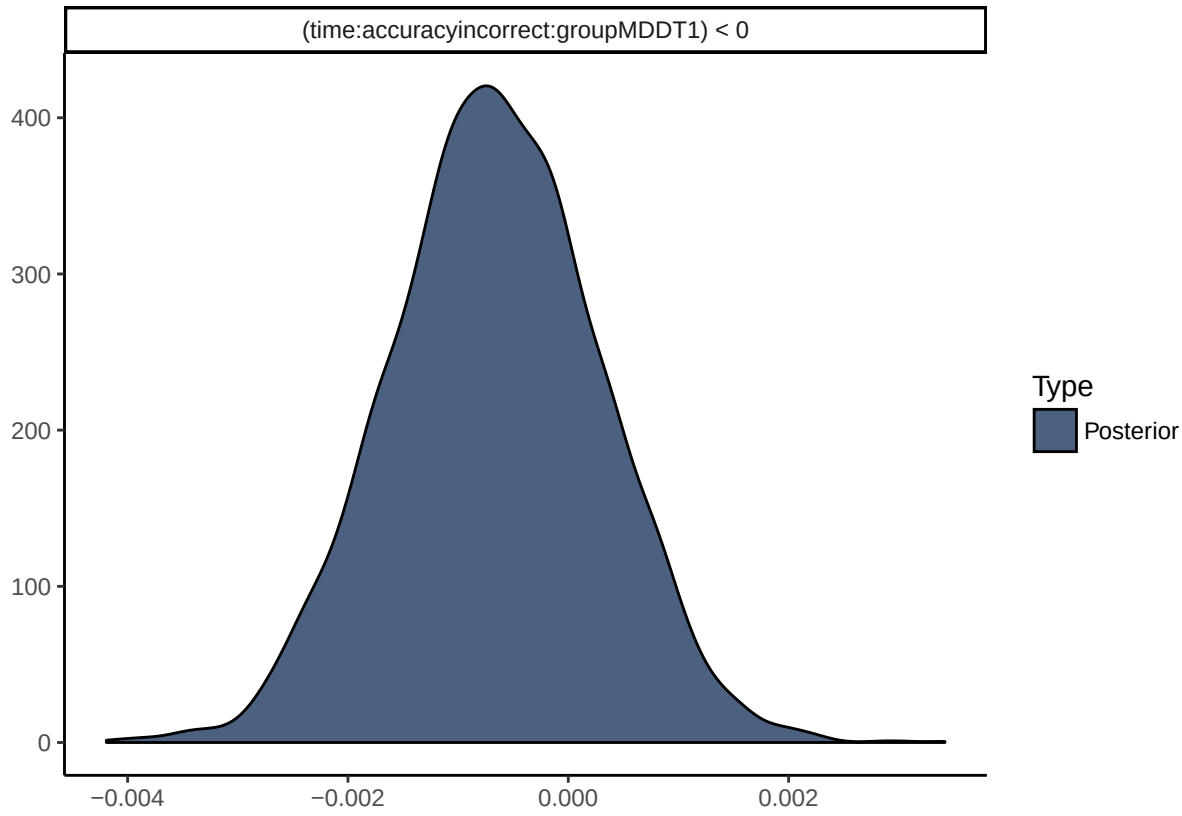
```
# Evid.Ratio is the ratio of P(time:accuracyincorrect < 0) / P(time:accuracyincorrect >= 0)
(hyp1 <- hypothesis(fit_timeaccuracy, "time:accuracyincorrect < 0"))
plot(hyp1)
```



Model 4: time*accuracy*group

2) Posterior probability that the time effect is steeper for incorrect vs correct in MDD vs HC

```
# Evid.Ratio is the ratio of P(time:accuracyincorrect:groupMDDT1 < 0) / P(time:accuracyincorrect:groupM
(hyp3 <- hypothesis(fit_timeaccuracygroup, "time:accuracyincorrect:groupMDDT1 < 0"))
plot(hyp3)
```


(time:accuracyincorrect:groupMDDT1) < 0
400
300
200
100
0
−0.004
−0.002
0.000
0.002
Type
Posterior