Analyzing data on the level of cognitive processes

Exercise 3 - Complex span task

In this exercise we will implement M3 models to experiments from Oberauer & Lewandowsky (2019) in which participants completed complex span tasks.

Task 1: Basic M3 model

In task 1, we will implement the basic M3 model that can be applied to the complex span task. We will use the data set from experiment 1 from Oberauer & Lewandowsky (2019). You can find this data set in the bmm package under the name of "oberauer_lewandowsky_2019_e1".

In this experiment, 40 participants completed a complex span task. They were asked to memorize lists of 5 words in order which was interleved by a distracting processing task. Participants were asked to read the presented word aloud. Participants selected their reported the memorized word by sleecting their responses from different displayed responses. The experiment consisted of three different conditions regarding the distractors in the processing task that each participant completed. In the control condition, there were always new distractors (*"new distractors"*) presented in the processing task. In the remaining conditions the distractors in the processing task matched the words in the memory task. These distractors either matched the order in the memory task so that the same word was presented in the processing task as in the memory task (*"old same"*) or the distractors were presented in a different order than the memory task words (*"old reordered"*).

Here is an overview what the individual variable in the data set refer to:

- ID: participant ID
- cond: distractor condition in the processing task (*"new distractors"*, *"old same"*, or *"old reordered"*)
- corr: number of correct responses (IIP)
- other: number of errors when a word from the list was reported, but at the incorrect position (IOP)
- npl: number of errors when a word *not* from the list was reported (NPL)
- dist: number of distractor words that were reported instead of a memory list word (only given in the *"new distractors"* condition)
- n_corr: total number of correct responses in each trial
- n_other: total number of IOP responses in each trial
- n_dist: total number of distractrs in each trial in the processing task
- n_npl: total number of NPL in each trial

We want to know now what influence the processing task had on memorizing the words and how different cognitive processes were impacted, namely binding memory and item memory. Specifically, how well did participants remember the memory list words when the same word was displayed in the processing task (*"old same"*) in comparison when it was not (*"old reordered"*).

To this end, we are interested in implementing in a first step the basic M3 model and we will focus on the conditions *"old reordered"* and *"old same"*. How are binding memory and item memory affected by the different distractor conditions? Does this match your expectations?

Load libraries

```
rm(list = ls())

library("here")
```

```
## here() starts at /Users/icoura/Desktop/Git/ws-process-level-data-analysis
```

```
library("bmm")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library("tidyr")
library("brms")
```

```
## Loading required package: Rcpp
```

```
## Loading 'brms' package (version 2.22.0). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').
```

```
##
## Attaching package: 'brms'
```

```
## The following object is masked from 'package:stats':
##
##     ar
```

```
library("ggplot2")
library("tidybayes")
```

```
##
## Attaching package: 'tidybayes'
```

```
## The following objects are masked from 'package:brms':
##
##     dstudent_t, pstudent_t, qstudent_t, rstudent_t
```

```
library("readr")
```

Read in data

```
mydata_exp1 <- oberauer_lewandowsky_2019_e1

mydata_exp1_filtered <- mydata_exp1 %>%
  filter(cond != "new distractors")
```

Implement basic M3 model

```
n_cores <- 4
n_iters <- 3000
```

```r
n_warmup <- 1000
n_chains <- 4

m3_model_ss <- m3(resp_cats = c("corr", "other", "npl"),
                  num_options = c("n_corr", "n_other", "n_npl"),
                  choice_rule = "softmax",
                  version = "ss")

m3_formula_ss <- bmf(
  corr ~ b + a + c,
  other ~ b + a,
  npl ~ b,
  c ~ 1 + cond + (1 + cond | ID),
  a ~ 1 + cond + (1 + cond | ID)
)

default_prior(m3_formula_ss, data = mydata_exp1_filtered, model = m3_model_ss)
```

```
## Duplicate parameter(s): 'corr', 'other', 'npl'. Overwriting the initial formula.
```

```
##                 prior class        coef group resp dpar nlpar   lb   ub
##                lkj(1)   cor
##                lkj(1)   cor                ID
##         normal(0,0.5)     b condoldsame                    a <NA> <NA>
##  student_t(3, 0, 2.5)    sd                                 a    0
##  student_t(3, 0, 2.5)    sd                ID               a    0
##  student_t(3, 0, 2.5)    sd condoldsame    ID               a    0
##  student_t(3, 0, 2.5)    sd   Intercept    ID               a    0
##                (flat)     b                                b
##           normal(0,2)     b condoldsame                    c <NA> <NA>
##  student_t(3, 0, 2.5)    sd                                 c    0
##  student_t(3, 0, 2.5)    sd                ID               c    0
##  student_t(3, 0, 2.5)    sd condoldsame    ID               c    0
##  student_t(3, 0, 2.5)    sd   Intercept    ID               c    0
##           normal(0,2)     b                                c <NA> <NA>
##           normal(3,1)     b   Intercept                    c <NA> <NA>
##         normal(0,0.5)     b                                a <NA> <NA>
##           normal(2,1)     b   Intercept                    a <NA> <NA>
##           constant(0)     b   Intercept                    b <NA> <NA>
##        source
##       default
##  (vectorized)
##  (vectorized)
##       default
##  (vectorized)
##  (vectorized)
##  (vectorized)
##       default
##  (vectorized)
##       default
##  (vectorized)
##  (vectorized)
##  (vectorized)
##          user
```

```
##          user
##          user
##          user
##          user
```

```
m3_fit_ss <- bmm(
  formula = m3_formula_ss,
  data = mydata_exp1_filtered,
  model = m3_model_ss,
  core = n_cores,
  chain = n_chains,
  iter = n_warmup + n_iters,
  warmup = n_warmup,
  file = here("models", "model_m3_ss"),
  file_refit = "on_change"
)
```

```
## Warning: The "on_change" option for the file_refit argument available in brms,
## is currently not implemented for bmm.
## To avoid overwriting an already saved bmmfit object, file_refit was set to "never".
```

```
summary(m3_fit_ss)
```

```
## Loading required package: rstan
```

```
## Warning: package 'rstan' was built under R version 4.4.3
```

```
## Loading required package: StanHeaders
```

```
## Warning: package 'StanHeaders' was built under R version 4.4.3
```

```
##
## rstan version 2.36.0.9000 (Stan version 2.37.0)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
```

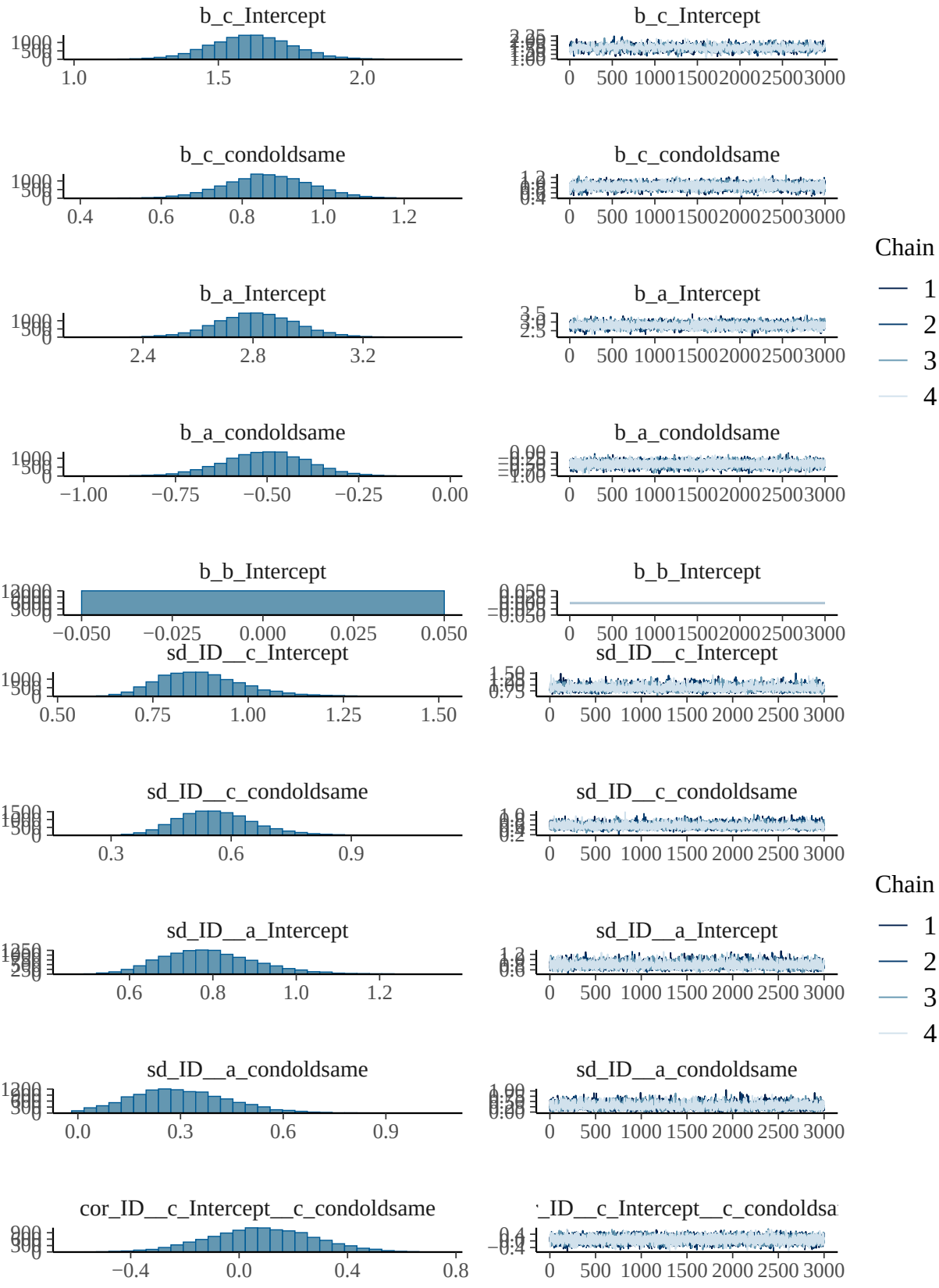
```
##
## Attaching package: 'rstan'
```

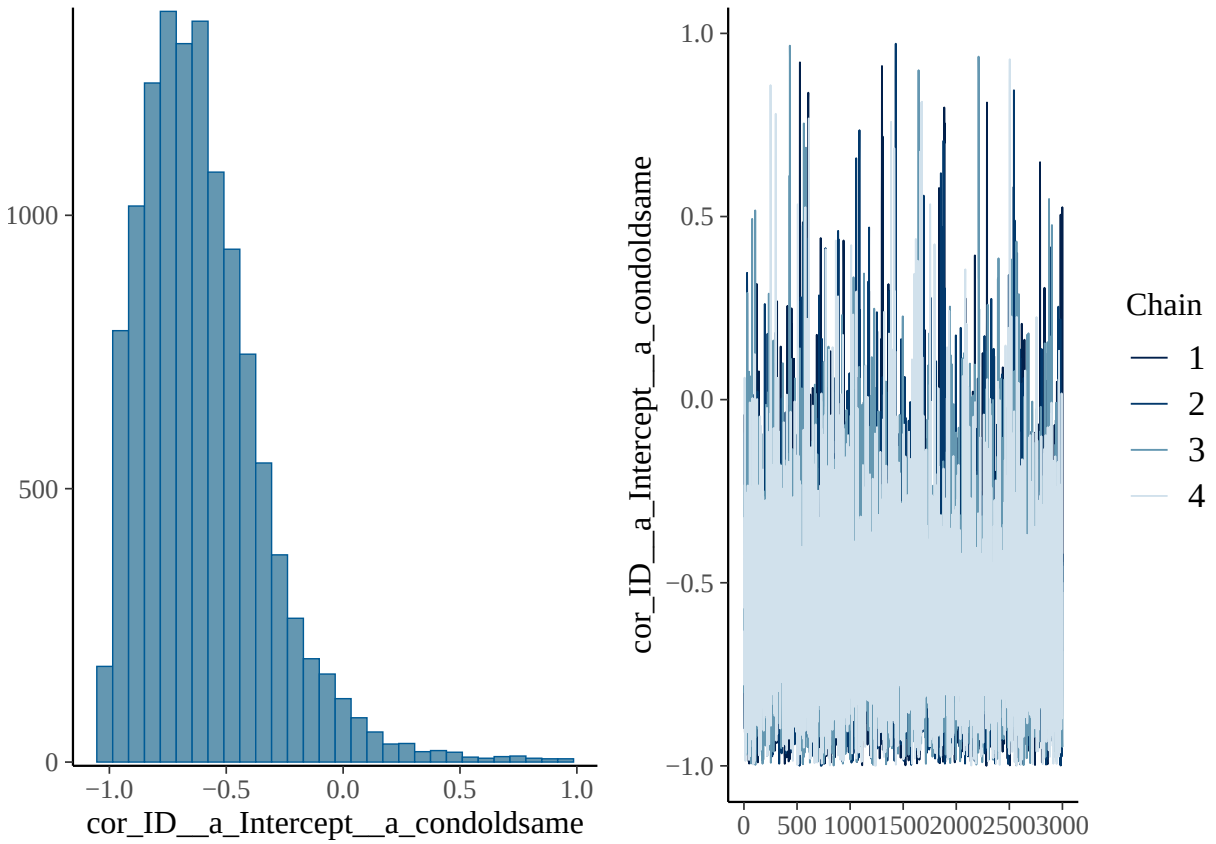
```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
##   Model: m3(resp_cats = c("corr", "other", "npl"),
##             num_options = c("n_corr", "n_other", "n_npl"),
##             choice_rule = "softmax",
##             version = "ss")
##   Links: c = identity; a = identity
## Formula: b = 0
##          c ~ 1 + cond + (1 + cond | ID)
##          a ~ 1 + cond + (1 + cond | ID)
##          corr ~ b + a + c
##          other ~ b + a
```

```
##          npl ~ b
##    Data: (Number of observations: 80)
##   Draws: 4 chains, each with iter = 4000; warmup = 1000; thin = 1;
##          total post-warmup draws = 12000
##
## Multilevel Hyperparameters:
## ~ID (Number of levels: 40)
##                                 Estimate Est.Error l-95% CI u-95% CI Rhat
## sd(c_Intercept)                     0.88      0.11     0.69     1.13 1.00
## sd(c_condoldsame)                   0.56      0.10     0.40     0.78 1.00
## sd(a_Intercept)                     0.79      0.12     0.59     1.05 1.00
## sd(a_condoldsame)                   0.30      0.14     0.04     0.61 1.00
## cor(c_Intercept,c_condoldsame)      0.08      0.19    -0.30     0.47 1.00
## cor(a_Intercept,a_condoldsame)     -0.60      0.27    -0.97     0.05 1.00
##                                 Bulk_ESS Tail_ESS
## sd(c_Intercept)                     2854     5115
## sd(c_condoldsame)                   4995     7302
## sd(a_Intercept)                     4056     7446
## sd(a_condoldsame)                   2757     3464
## cor(c_Intercept,c_condoldsame)      4589     6729
## cor(a_Intercept,a_condoldsame)      6564     4823
##
## Regression Coefficients:
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## c_Intercept       1.61      0.14     1.33     1.90 1.00     1356     2485
## c_condoldsame     0.86      0.11     0.65     1.07 1.00     4851     7049
## a_Intercept       2.81      0.14     2.53     3.10 1.00     3721     5845
## a_condoldsame    -0.50      0.12    -0.74    -0.28 1.00    10110     8622
##
## Constant Parameters:
##                 Value
## b_Intercept      0.00
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot(m3_fit_ss)
```

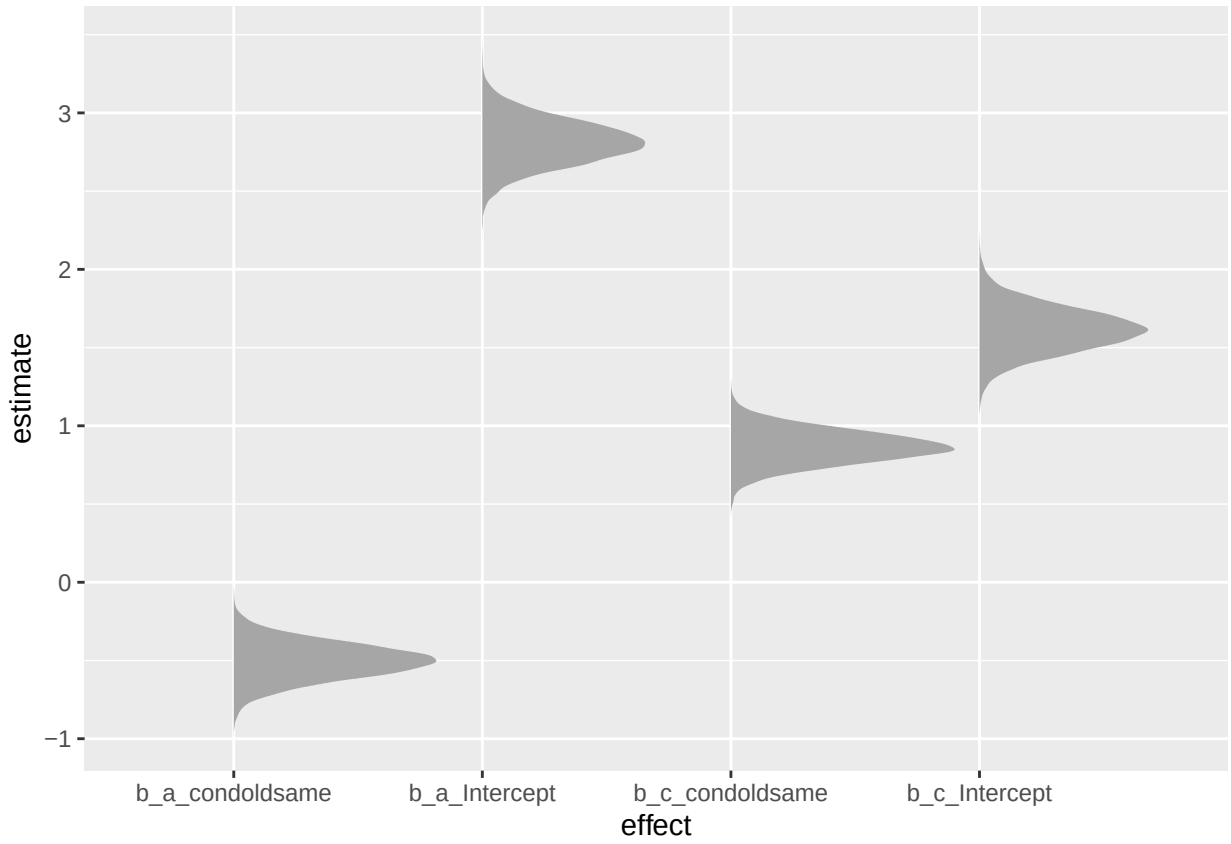

b_c_Intercept
b_c_Intercept
b_c_condoldsame
b_c_condoldsame
b_a_Intercept
b_a_Intercept
b_a_condoldsame
b_a_condoldsame
b_b_Intercept
b_b_Intercept
sd_ID__c_Intercept
sd_ID__c_Intercept
sd_ID__c_condoldsame
sd_ID__c_condoldsame
sd_ID__a_Intercept
sd_ID__a_Intercept
sd_ID__a_condoldsame
sd_ID__a_condoldsame
cor_ID__c_Intercept__c_condoldsame
_ID__c_Intercept__c_condoldsa
Chain
1
2
3
4
Chain
1
2
3
4




```
posterior <- as_draws_df(m3_fit_ss) %>%
  select(starts_with("b_")) %>%
  pivot_longer(cols = starts_with("b_"), names_to = "effect", values_to = "estimate") %>%
  filter(effect != "b_b_Intercept")
```

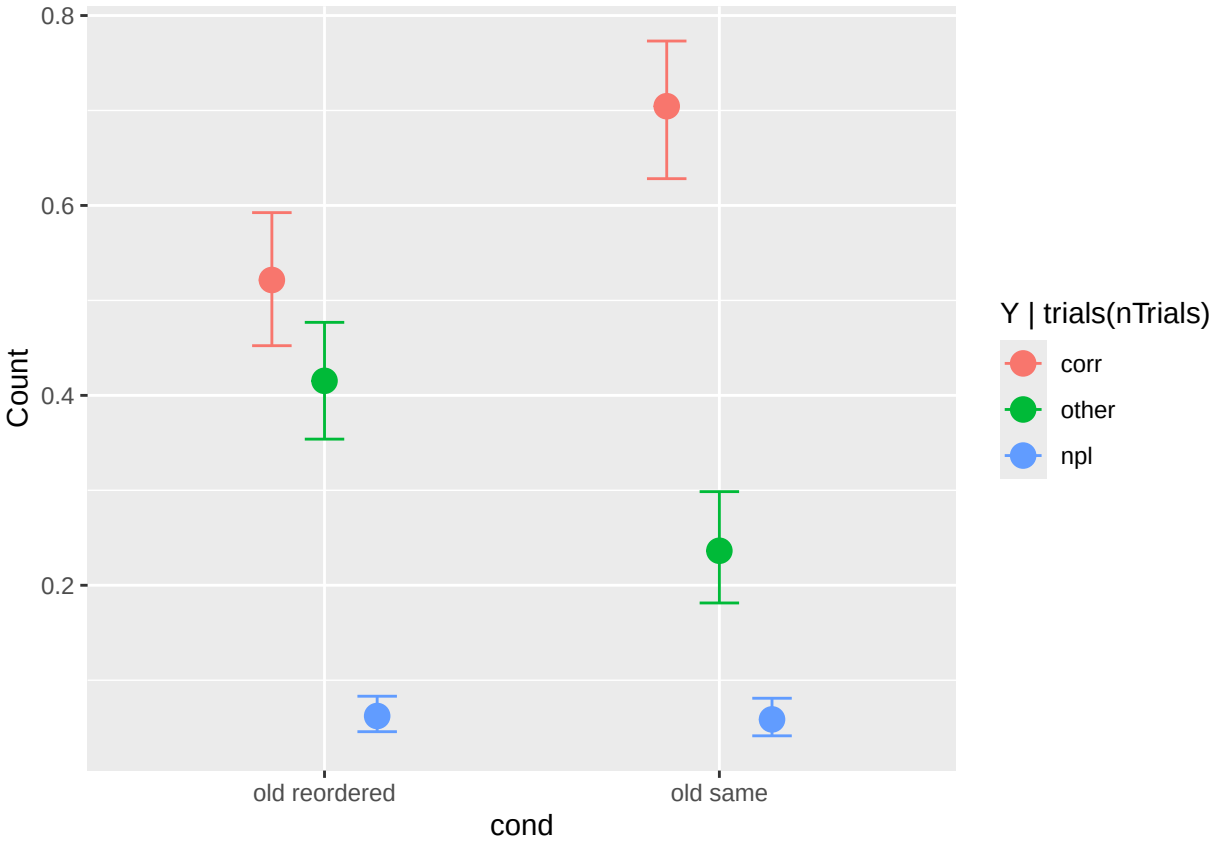
```
## Warning: Dropping 'draws_df' class as required metadata was removed.
```

```
ggplot(posterior, aes(x = effect, y = estimate)) +
  stat_slab()
```




```
conditional_effects(m3_fit_ss, categorical = T, effects = "cond")
```

```
## Setting all 'trials' variables to 1 by default if not specified otherwise.
```



Task 2: Extended M3 model

In task 2 we will now extend our M3 model to the complex span task. In order to do so we need more information about the errors that participants did regarding the distractors from the processing task. Therefore, we will use in this task the data set from experiment 2 from Oberauer & Lewandowsky (2019) in which the number of distractors in position (DIP) and distractors in other position (DOP) are reported as well. You can find this data set "data_m3_exp2" in the folder "data".

This data set now includes as well the following variables:

- condition: length of free time interval after each distractor, short (0.2 s; *"Low"*) or long (1.7 s; *"High"*) interval
- dip: number of errors when a distractor from the processing task was reported in the correct position as the memory item (DIP)
- dop: number of errors when a distractor from the processing task was reported, but from the incorrect position (DOP)

In this experiment, 27 participants completed a complex span task. Similarly to experiment 1, they memorized lists of 5 words in order while completing a distracting processing task that immediately followed and were not supposed to memorize. Yet, in experiment 2, participants were asked to judge whether the presented word is smaller or larger than a soccer ball instead of reading the word aloud. This applied to the memory as well as the processing task. They again reported the memory word by selecting from a pool of displayed words. The experiment consisted of two conditions regarding the free time interval after the processing task. The time interval after each distractor was either long (*"High"*) or short (*"Low"*).

In this task, we are interested in whether the free time interval (*"High"*) or (*"Low"*) after the distractor has an influence on the filtering of the distractors. Hence, we extend our M3 model for the complex span task to estimate as well the filtering process apart from the strength of binding and item memory. How is the

filtering parameter affected by the different distractor conditions? Does binding and item memory differ among the free time conditions?

```
mydata_exp2 <- read_csv(here("data", "data_m3_exp2.csv"))
```

```
## Rows: 54 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): condition
## dbl (6): ID, corr, other, dip, dop, npl
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Memdat <- read.table(here("data", "data_m3_exp2.dat"), header=FALSE, row.names=NULL)
# names(Memdat) <- c("NcorrHigh", "NcorrLow", "NotherHigh", "NotherLow", "NDinposHigh", "NDinposLow", "
#
# long_data <- Memdat %>%
#   pivot_longer(
#     cols = starts_with("N"),  # everything except id
#     names_to = c(".value", "condition"),  # split names into variable and condition
#     names_pattern = "(.+)(Low|High)"
#   )
#
# mydata_exp2 <- long_data %>%
#   rename(corr = Ncorr,
#          other = Nother,
#          dip = NDinpos,
#          dop = NDother,
#          npl = Nnpl)
#
# mydata_exp2 <- mydata_exp2 %>%
#   add_column(ID = rep(1:27, each = 2), .before = 1)
#
# write_csv(mydata_exp2, here("data", "data_m3_exp2.csv"))

mydata_exp2_filtered <- mydata_exp2 %>%
  filter(condition == "High")
```

```
m3_model_cs <- m3(resp_cats = c("corr", "other", "dip", "dop", "npl"),
                  num_options = c(1, 4, 1, 4, 5),
                  choice_rule = "softmax",
                  version = "cs")

m3_formula_cs <- bmf(
  corr ~ b + a + c,
  other ~ b + a,
  dip ~ b + f * (a + c),
  dop ~ b + f * a,
  npl ~ b,
  c ~ 1 + condition + (1 + condition | ID),
  a ~ 1 + condition + (1 + condition | ID),
  f ~ 1 + condition + (1 + condition | ID)
)
```

```
default_prior(m3_formula_cs, data = mydata_exp2, model = m3_model_cs)
```

```
## Duplicate parameter(s): 'corr', 'other', 'dip', 'dop', 'npl'. Overwriting the initial formula.
```

```
##                 prior class         coef group resp dpar nlpar   lb   ub
##                lkj(1)   cor
##                lkj(1)   cor                 ID
##         normal(0,0.5)     b conditionLow                    a <NA> <NA>
##  student_t(3, 0, 2.5)    sd                                 a    0
##  student_t(3, 0, 2.5)    sd                 ID              a    0
##  student_t(3, 0, 2.5)    sd conditionLow    ID              a    0
##  student_t(3, 0, 2.5)    sd    Intercept    ID              a    0
##                (flat)     b                                 b
##           normal(0,2)     b conditionLow                    c <NA> <NA>
##  student_t(3, 0, 2.5)    sd                                 c    0
##  student_t(3, 0, 2.5)    sd                 ID              c    0
##  student_t(3, 0, 2.5)    sd conditionLow    ID              c    0
##  student_t(3, 0, 2.5)    sd    Intercept    ID              c    0
##           normal(0,1)     b conditionLow                    f <NA> <NA>
##  student_t(3, 0, 2.5)    sd                                 f    0
##  student_t(3, 0, 2.5)    sd                 ID              f    0
##  student_t(3, 0, 2.5)    sd conditionLow    ID              f    0
##  student_t(3, 0, 2.5)    sd    Intercept    ID              f    0
##           normal(0,2)     b                                 c <NA> <NA>
##           normal(3,1)     b    Intercept                    c <NA> <NA>
##         normal(0,0.5)     b                                 a <NA> <NA>
##           normal(3,1)     b    Intercept                    a <NA> <NA>
##           normal(0,1)     b                                 f <NA> <NA>
##         logistic(0,1)     b    Intercept                    f <NA> <NA>
##           constant(0)     b    Intercept                    b <NA> <NA>
##        source
##       default
##  (vectorized)
##  (vectorized)
##       default
##  (vectorized)
##  (vectorized)
##  (vectorized)
##       default
##  (vectorized)
##       default
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##       default
##  (vectorized)
##  (vectorized)
##  (vectorized)
##          user
##          user
##          user
##          user
##          user
```

```
##          user
##          user
```

```
m3_fit_cs <- bmm(
  formula = m3_formula_cs,
  data = mydata_exp2,
  model = m3_model_cs,
  core = n_cores,
  chain = n_chains,
  iter = n_warmup + n_iters,
  warmup = n_warmup,
  file = here("models", "model_m3_cs"),
  file_refit = "on_change"
)
```

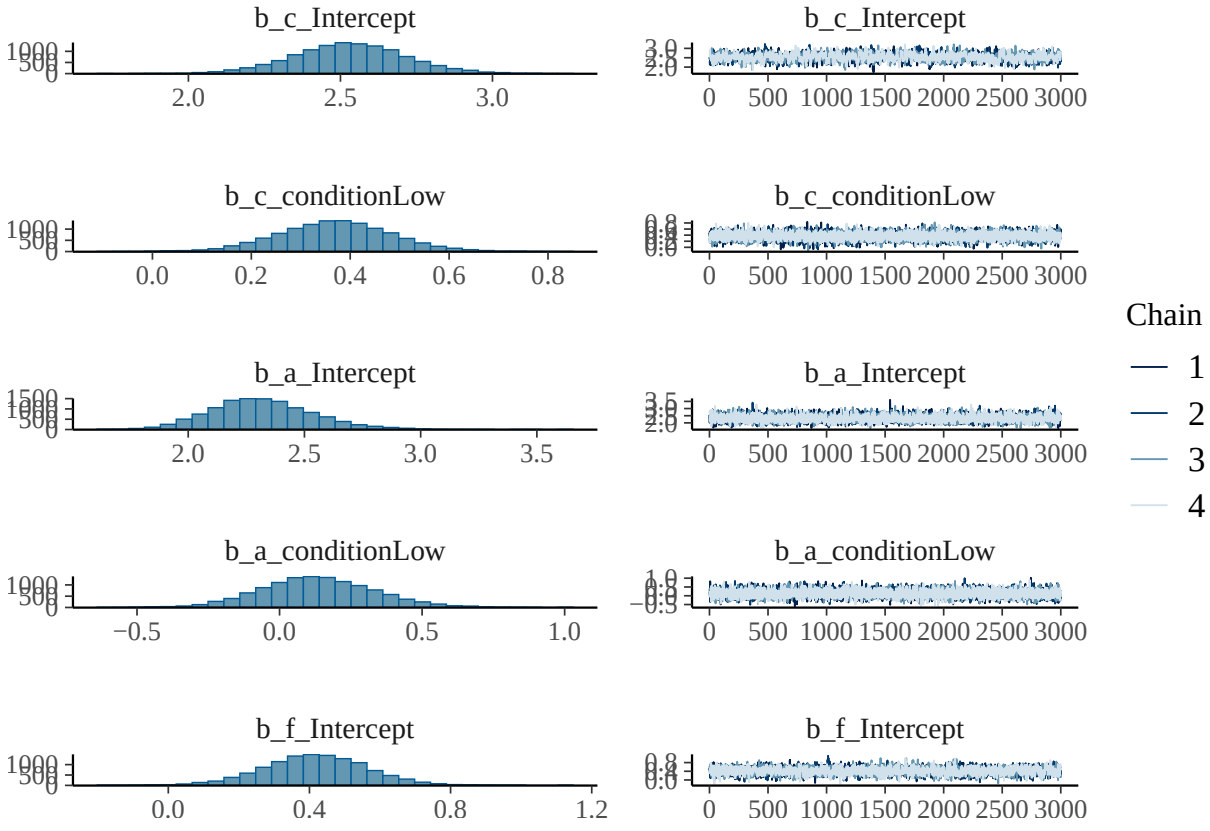
```
## Warning: The "on_change" option for the file_refit argument available in brms,
## is currently not implemented for bmm.
## To avoid overwriting an already saved bmmfit object, file_refit was set to "never".
```

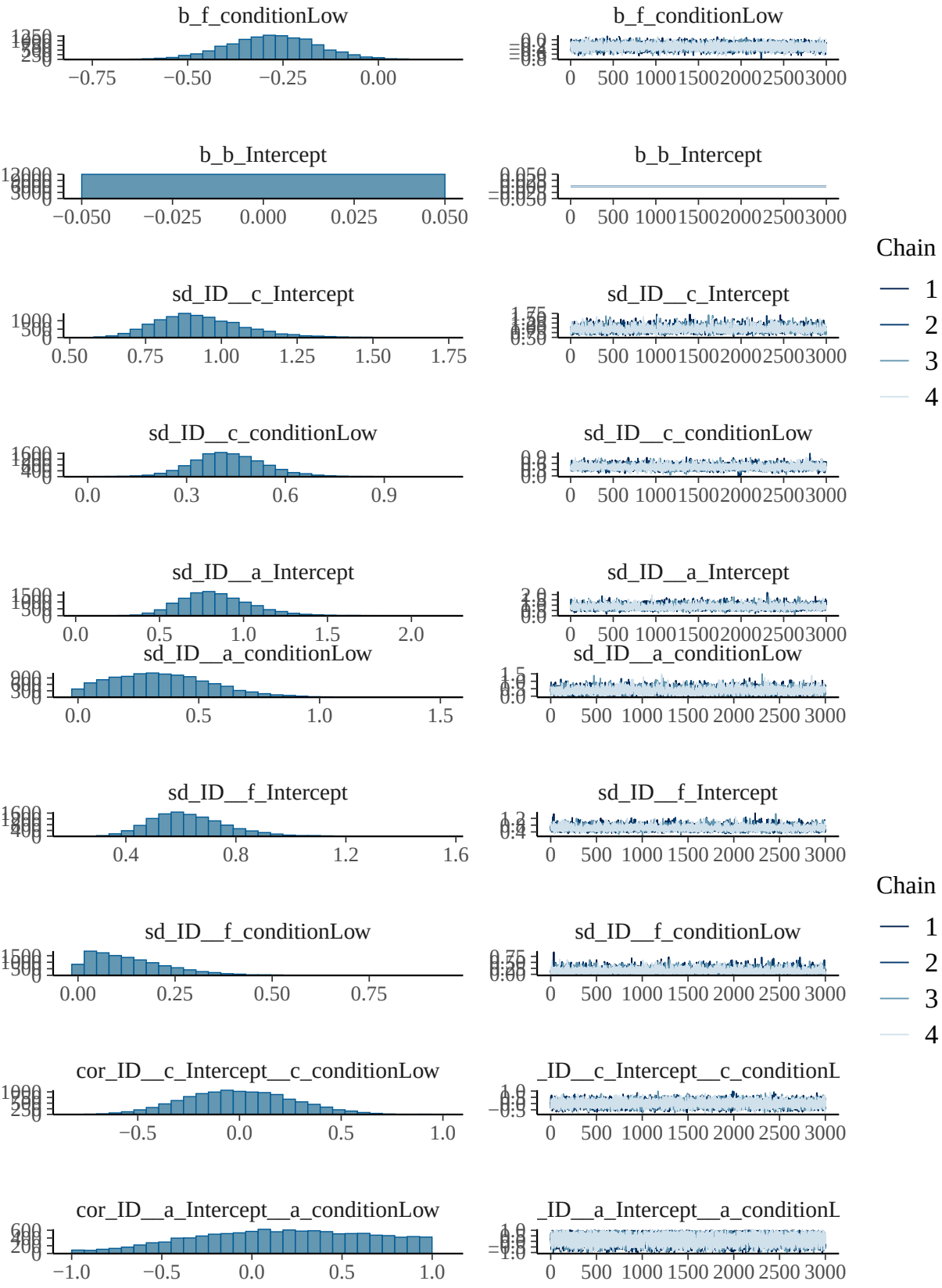
```
summary(m3_fit_cs)
```

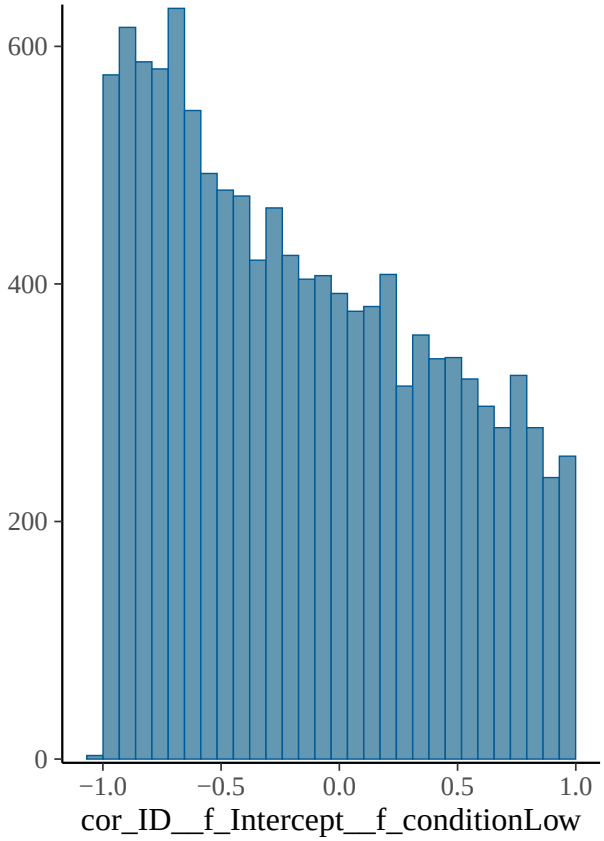
```
##   Model: m3(resp_cats = c("corr", "other", "dip", "dop", "npl"),
##             num_options = c(1, 4, 1, 4, 5),
##             choice_rule = "softmax",
##             version = "cs")
##   Links: c = identity; a = identity; f = logit
## Formula: b = 0
##          c ~ 1 + condition + (1 + condition | ID)
##          a ~ 1 + condition + (1 + condition | ID)
##          f ~ 1 + condition + (1 + condition | ID)
##          corr ~ b + a + c
##          other ~ b + a
##          dip ~ b + f * (a + c)
##          dop ~ b + f * a
##          npl ~ b
##    Data: (Number of observations: 54)
##   Draws: 4 chains, each with iter = 4000; warmup = 1000; thin = 1;
##          total post-warmup draws = 12000
##
## Multilevel Hyperparameters:
## ~ID (Number of levels: 27)
##                                 Estimate Est.Error l-95% CI u-95% CI Rhat
## sd(c_Intercept)                     0.93      0.15     0.69     1.28 1.00
## sd(c_conditionLow)                  0.43      0.11     0.22     0.67 1.00
## sd(a_Intercept)                     0.84      0.21     0.48     1.31 1.00
## sd(a_conditionLow)                  0.35      0.21     0.02     0.80 1.00
## sd(f_Intercept)                     0.63      0.14     0.40     0.95 1.00
## sd(f_conditionLow)                  0.13      0.10     0.01     0.38 1.00
## cor(c_Intercept,c_conditionLow)     0.00      0.26    -0.49     0.53 1.00
## cor(a_Intercept,a_conditionLow)     0.17      0.48    -0.78     0.95 1.00
## cor(f_Intercept,f_conditionLow)    -0.15      0.57    -0.96     0.92 1.00
##                                 Bulk_ESS Tail_ESS
## sd(c_Intercept)                     2852     5242
## sd(c_conditionLow)                  3565     2664
## sd(a_Intercept)                     4455     6852
```

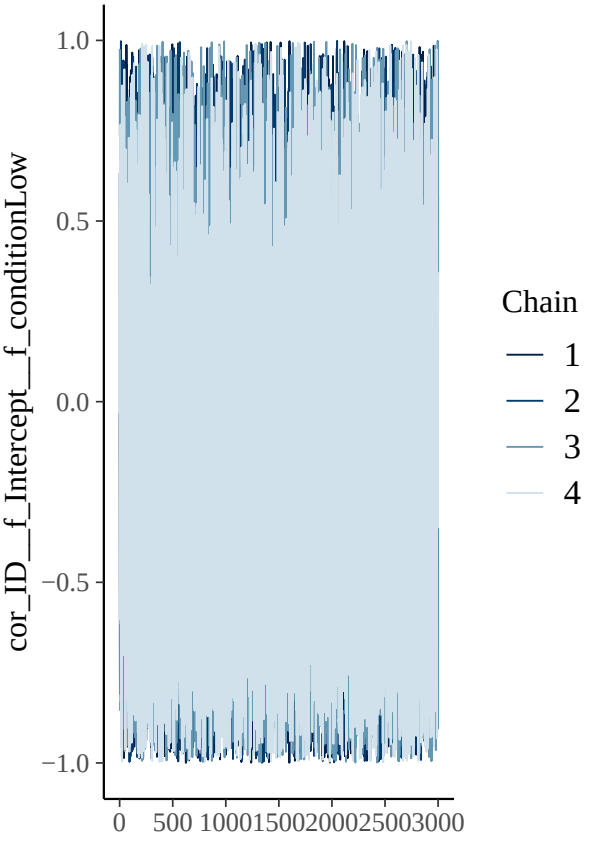
```
## sd(a_conditionLow)                    2801     4022
## sd(f_Intercept)                       4888     7813
## sd(f_conditionLow)                    5947     6136
## cor(c_Intercept,c_conditionLow)       6842     6379
## cor(a_Intercept,a_conditionLow)       6972     7406
## cor(f_Intercept,f_conditionLow)      12876     8345
##
## Regression Coefficients:
##                 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## c_Intercept         2.53      0.19     2.16     2.90 1.00     1558     2969
## c_conditionLow      0.37      0.12     0.14     0.60 1.00     7567     8185
## a_Intercept         2.32      0.22     1.91     2.79 1.00     4725     6763
## a_conditionLow      0.13      0.20    -0.25     0.54 1.00     9224     8367
## f_Intercept         0.42      0.15     0.12     0.70 1.00     3908     5967
## f_conditionLow     -0.28      0.12    -0.51    -0.05 1.00    10090     8376
##
## Constant Parameters:
##                  Value
## b_Intercept       0.00
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot(m3_fit_cs)
```




b_f_conditionLow
b_b_Intercept
sd_ID__c_Intercept
sd_ID__c_conditionLow
sd_ID__a_Intercept
sd_ID__a_conditionLow
sd_ID__f_Intercept
sd_ID__f_conditionLow
cor_ID__c_Intercept__c_conditionLow
cor_ID__a_Intercept__a_conditionLow
Chain
1
2
3
4

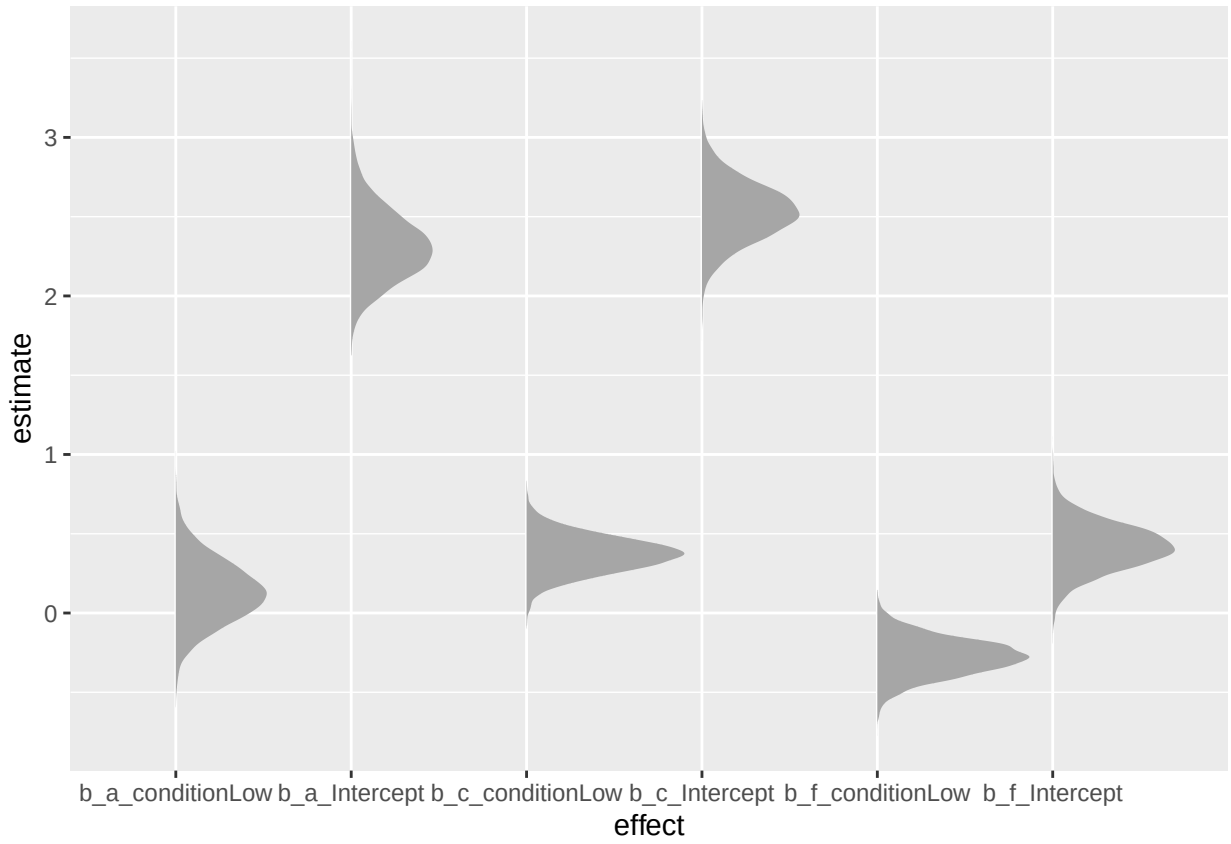





```
posterior <- as_draws_df(m3_fit_cs) %>%
  select(starts_with("b_")) %>%
  pivot_longer(cols = starts_with("b_"), names_to = "effect", values_to = "estimate") %>%
  filter(effect != "b_b_Intercept")
```

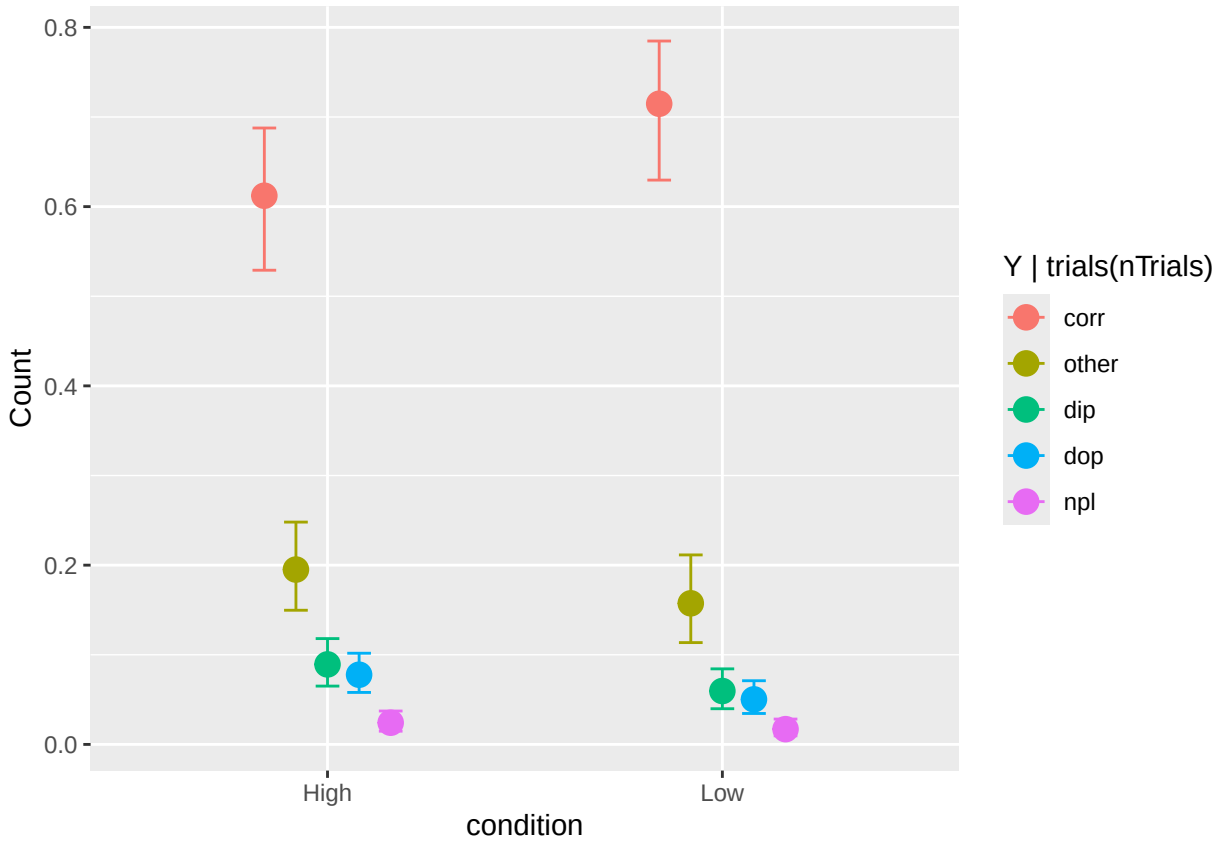
```
## Warning: Dropping 'draws_df' class as required metadata was removed.
```

```
ggplot(posterior, aes(x = effect, y = estimate)) +
  stat_slab()
```




```
conditional_effects(m3_fit_cs, categorical = T, effects = "condition")
```

```
## Setting all 'trials' variables to 1 by default if not specified otherwise.
```


0.8
0.6
0.4
0.2
0.0
Count
High
Low
condition
Y | trials(nTrials)
corr
other
dip
dop
npl