Tracing counterfactual development

October 11, 2020

Contents

Load packages

```
library("janitor")      # clean up column names
library("knitr")        # for knitting stuff
library("kableExtra")   # for markdown tables
library("xtable")       # for latex tables
library("broom")        # for tidy model fits
library("ggeffects")    # for marginal effects
library("BayesFactor")  # for calculating Bayes factors
library("TreeBUGS")     # for estimating the hierarchical Bayesian MPT model
library("tidybayes")    # for extracting parameters from Bayesian model
library("ggridges")     # for ridge line plot
library("MPTinR")       # for calculating MPT predictions
library("afex")         # for mixed model analysis of Exp. 2
library("patchwork")    # for making figure panels
library("tidyverse")    # everything else
```

```
theme_set(theme_classic() +
            theme(text = element_text(size = 24)))

opts_chunk$set(comment = "#>",
               fig.show = "hold")

options(dplyr.summarise.inform = F)
```

Helper functions

```
print_table = function(data, format = "html", digits = 2){
  if(format == "html"){
    data %>%
      kable(digits = digits) %>%
      kable_styling()
  }else if(format == "latex"){
    data %>%
      xtable(digits = digits) %>%
      print(include.rownames = F,
            booktabs = T)
  }
}
```

Experiment 1a

Read in data

```
df.exp1a = read_csv("../data/experiment1a_data.csv") %>%
  clean_names() %>%
  filter(age_group != 4) %>% # remove 4 year old children from pilot
  mutate(age_group = factor(age_group)) %>%
  rename(condition = cond)
```

Demographics

```
df.exp1a %>%
  filter(gng == 1) %>%
  group_by(age_group, sex) %>%
  summarize(count = n()) %>%
  group_by(age_group) %>%
  mutate(n = sum(count)) %>%
  pivot_wider(names_from = sex,
              values_from = count) %>%
  print_table()

df.exp1a %>%
  filter(gng == 0) %>%
  group_by(age_group, sex) %>%
  summarize(count = n()) %>%
  group_by(age_group) %>%
  mutate(n = sum(count)) %>%
  pivot_wider(names_from = sex,
              values_from = count) %>%
  print_table()
```

Stats

Overall logistic regression

```
fit = glm(formula = resp_acc ~ age_group * condition,
          family = "binomial",
          data = df.exp1a %>%
            filter(gng == 1))

fit %>%
  tidy() %>%
  print_table()
```

term	estimate	std.error	statistic	p.value
(Intercept)	2.94	1.03	2.87	0.00
age_group7	16.62	2404.67	0.01	0.99
age_group9	16.62	2404.67	0.01	0.99
conditionD	-2.54	1.12	-2.26	0.02
age_group7:conditionD	-17.23	2404.67	-0.01	0.99
age_group9:conditionD	-16.83	2404.67	-0.01	0.99

Logistic regression in overdetermined condition

```
fit = glm(formula = resp_acc ~ age_group,
          family = "binomial",
          data = df.exp1a %>%
            filter(gng == 1,
                   condition == "D"))

fit %>%
  tidy() %>%
```

```
  print_table()
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.41	0.46	0.89	0.37
age_group7	-0.61	0.64	-0.95	0.34
age_group9	-0.20	0.64	-0.32	0.75

Binomial exact test

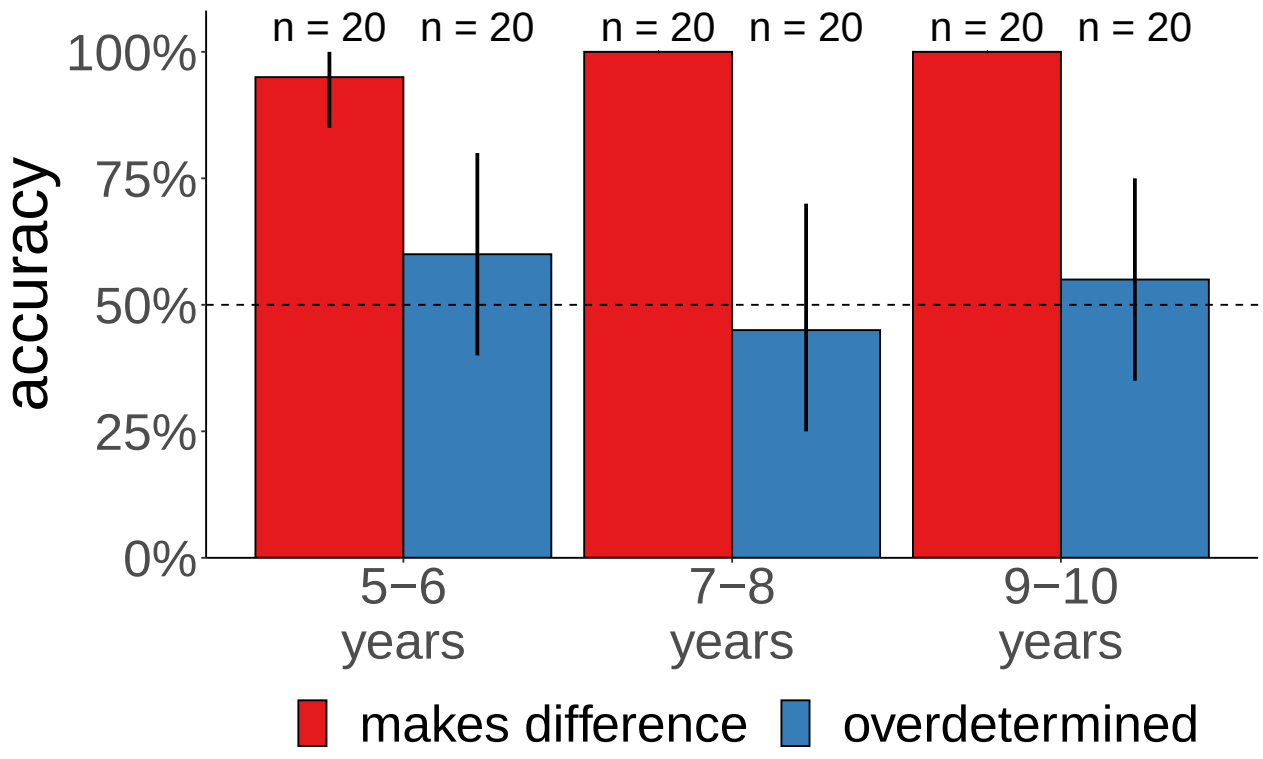
```
binom.test(x = df.exp1a %>%
             filter(gng == 1,
                    condition == "D") %>%
             summarize(sum = sum(resp_acc)) %>%
             pull(sum),
           n = df.exp1a %>%
             filter(gng == 1,
                    condition == "D") %>%
             nrow(.)) %>%
  tidy() %>%
  print_table()
```

estimate	statistic	p.value	parameter	conf.low	conf.high	method	alternative
0.53	32	0.7	60	0.4	0.66	Exact binomial test	two.sided

Plots

```
set.seed(1)

df.plot = df.exp1a %>%
  mutate(age_group = factor(age_group,
                            levels = c("5", "7", "9"),
                            labels = c("5-6\nyears",
                                       "7-8\nyears",
                                       "9-10\nyears")),
         condition = factor(condition,
                            levels = c("C", "D"),
                            labels = c("makes difference",
                                       "overdetermined"))) %>%
  filter(gng == 1)

df.text = df.plot %>%
  count(age_group, condition) %>%
  mutate(n = str_c("n = ", n),
         resp_acc = 1.05)

ggplot(data = df.plot,
       mapping = aes(x = age_group,
                     y = resp_acc,
                     group = condition,
                     fill = condition)) +
  stat_summary(fun = "mean",
```

```
               geom = "bar",
               color = "black",
               position = position_dodge(width = 0.9)) +
  stat_summary(fun.data = "mean_cl_boot",
               geom = "linerange",
               color = "black",
               position = position_dodge(width = 0.9),
               size = 1) +
  geom_text(data = df.text,
            mapping = aes(label = n),
            position = position_dodge(width = 0.9),
            size = 8) +
  geom_hline(yintercept = 0.5,
             linetype = 2) +
  labs(x = "age group",
       y = "accuracy") +
  scale_fill_brewer(palette = "Set1") +
  scale_y_continuous(breaks = seq(0, 1, 0.25),
                     labels = str_c(seq(0, 100, 25), "%"),
                     expand = expansion(mult = c(0, .03))) +
  theme(text = element_text(size = 36),
        legend.position = "bottom",
        axis.title.x = element_blank(),
        legend.title = element_blank())

ggsave("../figures//experiment1a_bars.pdf",
       width = 10,
       height = 6)
```



Experiment 1b

Read in data

```
df.exp1b = read_csv("../data/experiment1b_data.csv") %>%
  clean_names() %>%
  filter(gng == 1)
```

Demographics

```
df.exp1b %>%
  count(age_group) %>%
  print_table()
```

age_group	n
5	21
7	26

Stats

```
df.exp1b %>%
  group_by(age_group) %>%
  summarize(n = n(),
            n_correct_disjunctive = sum(disjunctive),
            n_correct_conjunctive = sum(disjunctive)) %>%
  print_table()
```

age_group	n	n_correct_disjunctive	n_correct_conjunctive
5	21	21	21
7	26	26	26

Experiment 2

Read in data

```
df.exp2 = read_csv("../data/experiment2_data.csv") %>%
  clean_names() %>%
  select(-contains("question")) %>%
  pivot_longer(cols = x1_1v1:x2_3,
               names_to = "question",
               values_to = "answer") %>%
  mutate(question = str_remove_all(question,"x"),
         answer = factor(answer,
                         levels = c("C", "DM", "ME", "DI"),
                         labels = c("correct", "match origin",
                                    "match trajectory", "match neither")),
         question = factor(question,
                           levels = c("1_1v1", "1_1v2", "1_2", "1_3",
                                      "2_1v1", "2_1v2", "2_2", "2_3")),
         gender = tolower(gender)) %>%
  rename(age_group = years) %>%
```

```
  separate(question, into = c("outcome", "type"), remove = F) %>%
  mutate(outcome_actual = factor(outcome, levels = 1:2,
                                 labels = c("goal", "miss")),
         outcome_counterfactual =
           ifelse(type %in% c("1v1", "1v2", "1v1", "1v2"), "same", "different"),
         collision = ifelse(type == "3", "no collision", "collision")) %>%
  filter(included == 1)
```

Demographics

```
df.exp2 %>%
  distinct(participant, gender, age_group) %>%
  count(age_group, gender, name = "count") %>%
  group_by(age_group) %>%
  mutate(n = sum(count)) %>%
  pivot_wider(names_from = gender,
              values_from = count) %>%
  print_table()
```

age_group	n	f	m
4	24	15	9
5	24	7	17
6	24	8	16

Stats

Performance on no-collision cases

```
df.exp2 %>%
  filter(collision == "no collision") %>%
  group_by(age_group) %>%
  count(answer) %>%
  group_by(age_group) %>%
  mutate(proportion = n / sum(n)) %>%
  filter(answer == "correct") %>%
  select(age_group, proportion) %>%
  print_table()
```

age_group	proportion
4	0.50
5	0.71
6	0.88

Mixed Model Accuracy Analysis

To analyse accuracy as a function of age group, the actual outcome, and the counterfactual outcome, we first need to prepare a new data set.

```
df2 = df.exp2 %>%
  filter(collision == "collision") %>%
  mutate(age_group = factor(age_group)) %>%
  group_by(participant, age_group, outcome_actual, outcome_counterfactual) %>%
```

```
  summarise(correct = sum(answer == "correct"),
            n = n()) %>%
  mutate(prop = correct/n)
```

We then fit a series of models (or read the fitted model files). `mod0` is the maximal model, `mod3` is the final model. To sum the analysis strategy up: Tests of fixed-effects were based on likelihood-ratio tests implemented in `afex`. We started this analysis using a model with the maximal random effect structure justified by the design (Barr et al., 2013). This model included by-participant random intercepts as well as by-participant random slopes for the actual and counterfactual outcome as well as their interaction. Because this model failed to converge, we reduced the random effect structure to a model with by-participant random intercepts only. For all tested models, including the maximal and the final model, the pattern of significant and non-significant effects was identical.

```
if (!file.exists("cache/mixed_exp2.rda")) {
  mod0 = mixed(prop ~ age_group*outcome_actual*outcome_counterfactual +
                 (outcome_actual * outcome_counterfactual||participant),
               df2, family = "binomial", method = "LRT", weights = n,
               expand_re = TRUE,
               control = glmerControl(optCtrl = list(maxfun = 1e5)))
  mod1 = mixed(prop ~ age_group*outcome_actual*outcome_counterfactual +
                 (outcome_actual + outcome_counterfactual||participant),
               df2, family = "binomial", method = "LRT", weights = n,
               expand_re = TRUE)
  mod2 = mixed(prop ~ age_group*outcome_actual*outcome_counterfactual +
                 (outcome_actual||participant),
               df2, family = "binomial", method = "LRT", weights = n,
               expand_re = TRUE)
  mod3 = mixed(prop ~ age_group*outcome_actual*outcome_counterfactual +
                 (1|participant),
               df2, family = "binomial", method = "LRT", weights = n,
               expand_re = TRUE)

  save(mod0, mod1, mod2, mod3, file = "fits/mixed_exp2.rda",
       compress = "xz")
} else {
  load("cache/mixed_exp2.rda")
}
```

The full model shows a main effect of age group plus an interaction of age group with the counterfactual outcome, but also some convergence warnings (for the full model).

```
mod0
```

```
#> Warning: lme4 reported (at least) the following warnings for 'full':
#>   * Model failed to converge with max|grad| = 0.0181666 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'age_group':
#>   * Model failed to converge with max|grad| = 0.012805 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'outcome_actual':
#>   * Model failed to converge with max|grad| = 0.0466433 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'outcome_counterfactual':
#>   * Model failed to converge with max|grad| = 0.0417475 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'age_group:outcome_actual':
#>   * Model failed to converge with max|grad| = 0.0852291 (tol = 0.002, component 1)
```

```
#> Warning: lme4 reported (at least) the following warnings for 'age_group:outcome_counterfactual':
#>   * Model failed to converge with max|grad| = 0.0635005 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'outcome_actual:outcome_counterfactual'
#>   * Model failed to converge with max|grad| = 0.0737963 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'age_group:outcome_actual:outcome_counte
#>   * Model failed to converge with max|grad| = 0.0478644 (tol = 0.002, component 1)

#> Mixed Model Anova Table (Type 3 tests, LRT-method)
#>
#> Model: prop ~ age_group * outcome_actual * outcome_counterfactual +
#> Model:     (outcome_actual * outcome_counterfactual || participant)
#> Data: df2
#> Df full model: 16
#>                                              Effect df   Chisq p.value
#> 1                                         age_group  2  8.80 *    .012
#> 2                                   outcome_actual  1    0.44    .509
#> 3                            outcome_counterfactual  1    1.20    .272
#> 4                          age_group:outcome_actual  2    4.57    .102
#> 5                 age_group:outcome_counterfactual  2 9.52 **    .009
#> 6             outcome_actual:outcome_counterfactual  1    1.17    .279
#> 7 age_group:outcome_actual:outcome_counterfactual  2    0.48    .786
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1
```

We consequently remove the random slope for the interaction. This model shows the same pattern of results, but also convergence warnings:

```
mod1
```

```
#> Warning: lme4 reported (at least) the following warnings for 'full':
#>   * Model failed to converge with max|grad| = 0.0146004 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'age_group':
#>   * boundary (singular) fit: see ?isSingular

#> Warning: lme4 reported (at least) the following warnings for 'outcome_actual':
#>   * Model failed to converge with max|grad| = 0.0261344 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'outcome_counterfactual':
#>   * Model failed to converge with max|grad| = 0.0621988 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'age_group:outcome_actual':
#>   * Model failed to converge with max|grad| = 0.0661235 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'age_group:outcome_counterfactual':
#>   * Model failed to converge with max|grad| = 0.0578913 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'outcome_actual:outcome_counterfactual'
#>   * Model failed to converge with max|grad| = 0.0362014 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'age_group:outcome_actual:outcome_counte
#>   * Model failed to converge with max|grad| = 0.022484 (tol = 0.002, component 1)

#> Mixed Model Anova Table (Type 3 tests, LRT-method)
#>
#> Model: prop ~ age_group * outcome_actual * outcome_counterfactual +
#> Model:     (outcome_actual + outcome_counterfactual || participant)
#> Data: df2
#> Df full model: 15
```

```
#>                                          Effect df   Chisq p.value
#> 1                                     age_group  2  8.80 *    .012
#> 2                                 outcome_actual  1    0.44    .509
#> 3                           outcome_counterfactual  1    1.20    .272
#> 4                        age_group:outcome_actual  2    4.57    .102
#> 5                  age_group:outcome_counterfactual  2 9.52 **    .009
#> 6            outcome_actual:outcome_counterfactual  1    1.17    .280
#> 7 age_group:outcome_actual:outcome_counterfactual  2    0.48    .786
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1
```

Inspection of the variance estimates shows that the one for the counterfactual outcome is smallest.

```
summary(mod1)$varcor
```

```
#>  Groups        Name                        Std.Dev.
#>  participant   (Intercept)                 1.8854359
#>  participant.1 re1.outcome_actual1         0.0042371
#>  participant.2 re1.outcome_counterfactual1 0.0028784
```

We remove this in the next step, but the model still shows convergence warnings:

```
mod2
```

```
#> Warning: lme4 reported (at least) the following warnings for 'full':
#>   * Model failed to converge with max|grad| = 0.0510619 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'age_group':
#>   * Model failed to converge with max|grad| = 0.0108255 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'outcome_actual':
#>   * Model failed to converge with max|grad| = 0.0301588 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'outcome_counterfactual':
#>   * Model failed to converge with max|grad| = 0.00309515 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'age_group:outcome_actual':
#>   * Model failed to converge with max|grad| = 0.0442321 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'age_group:outcome_counterfactual':
#>   * Model failed to converge with max|grad| = 0.0472595 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'outcome_actual:outcome_counterfactual'
#>   * Model failed to converge with max|grad| = 0.0600451 (tol = 0.002, component 1)

#> Warning: lme4 reported (at least) the following warnings for 'age_group:outcome_actual:outcome_counte
#>   * Model failed to converge with max|grad| = 0.0124133 (tol = 0.002, component 1)

#> Mixed Model Anova Table (Type 3 tests, LRT-method)
#>
#> Model: prop ~ age_group * outcome_actual * outcome_counterfactual +
#> Model:     (outcome_actual || participant)
#> Data: df2
#> Df full model: 14
#>                                          Effect df   Chisq p.value
#> 1                                     age_group  2  8.79 *    .012
#> 2                                 outcome_actual  1    0.43    .510
#> 3                           outcome_counterfactual  1    1.20    .273
#> 4                        age_group:outcome_actual  2    4.57    .102
#> 5                  age_group:outcome_counterfactual  2 9.59 **    .008
#> 6            outcome_actual:outcome_counterfactual  1    1.17    .280
```

```
#> 7 age_group:outcome_actual:outcome_counterfactual  2    0.48    .787
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1
```

We consequently remove the last remaining random slope as well. This still shows some warning, but no more for the full model. The pattern of results remains the same.

```
mod3
```

```
#> Warning: lme4 reported (at least) the following warnings for 'outcome_actual':
#>   * Model failed to converge with max|grad| = 0.0058044 (tol = 0.002, component 1)
```

```
#> Warning: lme4 reported (at least) the following warnings for 'age_group:outcome_actual':
#>   * Model failed to converge with max|grad| = 0.00416424 (tol = 0.002, component 1)
```

```
#> Warning: lme4 reported (at least) the following warnings for 'age_group:outcome_counterfactual':
#>   * Model failed to converge with max|grad| = 0.0108411 (tol = 0.002, component 1)
```

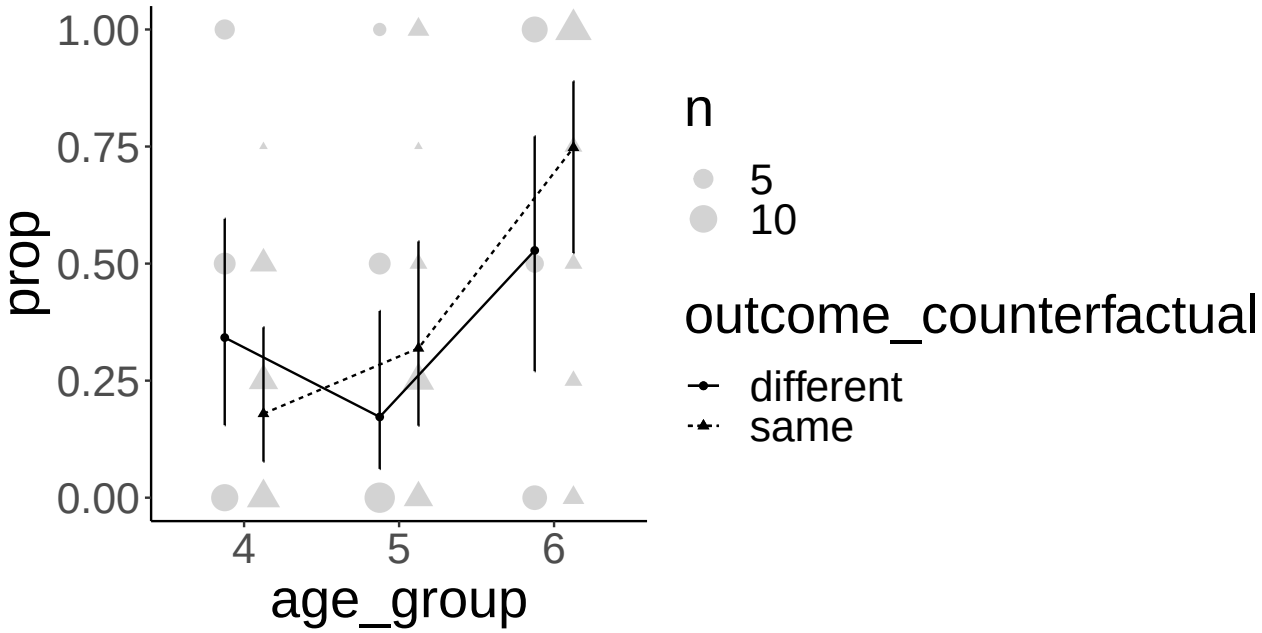
```
#> Mixed Model Anova Table (Type 3 tests, LRT-method)
#>
#> Model: prop ~ age_group * outcome_actual * outcome_counterfactual +
#> Model:     (1 | participant)
#> Data: df2
#> Df full model: 13
#>                                           Effect df   Chisq p.value
#> 1                                      age_group  2  8.80 *    .012
#> 2                                 outcome_actual  1    0.44    .509
#> 3                        outcome_counterfactual  1    1.20    .273
#> 4                       age_group:outcome_actual  2    4.57    .102
#> 5              age_group:outcome_counterfactual  2 9.59 **    .008
#> 6         outcome_actual:outcome_counterfactual  1    1.17    .280
#> 7 age_group:outcome_actual:outcome_counterfactual  2    0.48    .786
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1
```

We can now investigate the interaction of age group and the counterfactual outcome. For this we use the final model.

```
afex_plot(mod3, "age_group", "outcome_counterfactual",
          data_geom = geom_count)
```

```
#> Aggregating data over: participant
```

```
#> NOTE: Results may be misleading due to involvement in interactions
```



This shows an age effect if the counterfactual outcome is the same (i.e., over-determined cases), but not if it is different (i.e., singly-determined cases). This is also confirmed by follow-up tests. First for the final modeL

```
pairs(emmeans::emmeans(mod3, by = "outcome_counterfactual", "age_group"))
```

```
#> NOTE: Results may be misleading due to involvement in interactions
```

```
#> outcome_counterfactual = different:
#>  contrast estimate    SE  df z.ratio p.value
#>  4 - 5       0.914 0.789 Inf  1.159  0.4780
#>  4 - 6      -0.767 0.776 Inf -0.988  0.5842
#>  5 - 6      -1.681 0.818 Inf -2.055  0.0994
#>
#> outcome_counterfactual = same:
#>  contrast estimate    SE  df z.ratio p.value
#>  4 - 5      -0.763 0.684 Inf -1.115  0.5048
#>  4 - 6      -2.607 0.717 Inf -3.637  0.0008
#>  5 - 6      -1.844 0.706 Inf -2.610  0.0246
#>
#> Results are averaged over the levels of: outcome_actual
#> Results are given on the log odds ratio (not the response) scale.
#> P value adjustment: tukey method for comparing a family of 3 estimates
```

And for the maximal model:

```
pairs(emmeans::emmeans(mod0, by = "outcome_counterfactual", "age_group"))
```

```
#> NOTE: Results may be misleading due to involvement in interactions
```

```
#> outcome_counterfactual = different:
#>  contrast estimate    SE  df z.ratio p.value
#>  4 - 5       0.911 0.789 Inf  1.154  0.4810
#>  4 - 6      -0.768 0.776 Inf -0.990  0.5834
#>  5 - 6      -1.679 0.818 Inf -2.051  0.1002
#>
#> outcome_counterfactual = same:
#>  contrast estimate    SE  df z.ratio p.value
```

```
#>  4 - 5      -0.763 0.685 Inf -1.115  0.5049
#>  4 - 6      -2.608 0.717 Inf -3.635  0.0008
#>  5 - 6      -1.844 0.707 Inf -2.608  0.0247
#>
#> Results are averaged over the levels of: outcome_actual
#> Results are given on the log odds ratio (not the response) scale.
#> P value adjustment: tukey method for comparing a family of 3 estimates
```

Do we see an effect of "wishful thinking" such that children in our study were more likely to be correct on trials for which the ball could have gone into the goal but did not (i.e., an actual outcome by counterfactual outcome interaction)? No.

```
emmeans::emmeans(mod3, c("outcome_actual", "outcome_counterfactual"),
                 type = "response")
```

```
#> NOTE: Results may be misleading due to involvement in interactions
```

```
#>  outcome_actual outcome_counterfactual  prob     SE  df asymp.LCL asymp.UCL
#>  goal           different              0.344 0.0875 Inf     0.197     0.529
#>  miss           different              0.318 0.0880 Inf     0.174     0.508
#>  goal           same                   0.346 0.0735 Inf     0.218     0.500
#>  miss           same                   0.461 0.0809 Inf     0.311     0.618
#>
#> Results are averaged over the levels of: age_group
#> Confidence level used: 0.95
#> Intervals are back-transformed from the logit scale
```

MPT analysis

We first need to prepare the data for fitting.

```
age_groups = c(4, 5, 6)
di = df.exp2 %>%
  filter(collision == "collision") %>% # Filter out no-collisions
  mutate(answer = as.numeric(answer)) %>%
  group_by(participant, months, age_group, answer) %>%
  summarise(count=n()) %>%
  spread(answer,count, fill = 0) %>%
  ungroup %>%
  mutate(age2 = as.numeric(substr(age_group, 1, 1)) + months/12) %>%
  mutate(age = factor(substr(age_group, 1, 1), levels = age_groups))

dc = di %>%
  select(age) %>%
  as_tibble()
```

We then fit the full model. Specifically, we fit one model to the data of all age groups, but add `age_group` as a covariate to all three model parameters. This is somewhat time intensive so we save the results (or load the saved results if those exist).

```
if (!file.exists("cache/fit_latent-trait_all.rda")) {
  fit_bayes_all = traitMPT("models/model.eqn",
                           di,
                           restrictions = list("g1=g2=0.5"),
                           covData = dc,
                           predStructure = list("m_o m_t s ; age"),
                           predType = "f",
```

```
                            n.iter = 320000,
                            n.thin = 300,
                            n.chains = 4,
                            n.adapt = 100000,
                            n.burnin = 20000)
  ppps = PPP(fit_bayes_all)
  save(fit_bayes_all, ppps, file = "fits/fit_latent-trait_all.rda",
       compress = "xz")
} else {
  load("cache/fit_latent-trait_all.rda")
}
```

We first check for chain convergence (i.e., Rhat and neff) and get an overview:

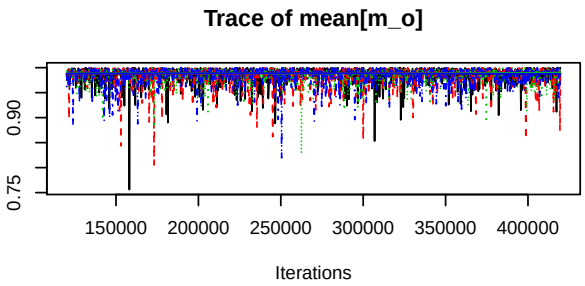
```
summary(fit_bayes_all)
```

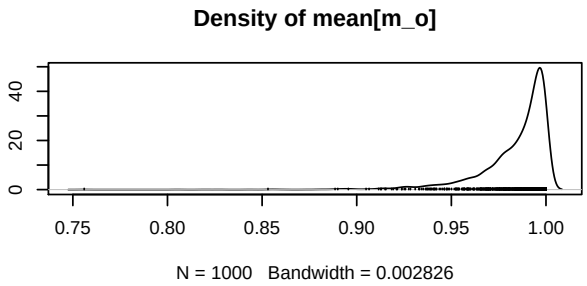
```
#> Call:
#> traitMPT(eqnfile = "model.eqn", data = di, restrictions = list("g1=g2=0.5"),
#>     covData = dc, predStructure = list("m_o m_t s ; age"), predType = "f",
#>     n.iter = 320000, n.adapt = 1e+05, n.burnin = 20000, n.thin = 300,
#>     n.chains = 4)
#>
#> Group-level medians of MPT parameters (probability scale):
#>           Mean    SD  2.5%   50% 97.5% Time-series SE n.eff  Rhat R_95%
#> mean_m_o 0.984 0.019 0.932 0.991 1.000          0.000  3360 1.004 1.006
#> mean_m_t 0.550 0.261 0.062 0.575 0.966          0.004  3691 1.002 1.006
#> mean_s   0.540 0.105 0.306 0.550 0.720          0.002  4007 1.001 1.002
#>
#> Mean/Median of latent-trait values (probit-scale) across individuals:
#>                Mean    SD   2.5%   50% 97.5% Time-series SE n.eff  Rhat R_95%
#> latent_mu_m_o 2.390 0.523  1.489 2.348 3.545          0.009  3628 1.000 1.000
#> latent_mu_m_t 0.163 0.858 -1.542 0.190 1.820          0.014  3674 1.002 1.007
#> latent_mu_s   0.103 0.276 -0.507 0.126 0.583          0.004  4007 1.001 1.002
#>
#> Standard deviation of latent-trait values (probit scale) across individuals:
#>                   Mean    SD  2.5%   50%  97.5% Time-series SE n.eff  Rhat
#> latent_sigma_m_o 1.232 0.724 0.092 1.173  2.805          0.012  3512 1.002
#> latent_sigma_m_t 7.558 4.070 3.039 6.566 17.783          0.094  1879 1.018
#> latent_sigma_s   1.479 0.516 0.741 1.389  2.774          0.009  3629 1.000
#>                  R_95%
#> latent_sigma_m_o 1.006
#> latent_sigma_m_t 1.030
#> latent_sigma_s   1.001
#>
#> Correlations of latent-trait values on probit scale:
#>                Mean    SD   2.5%    50% 97.5% Time-series SE n.eff  Rhat R_95%
#> rho[m_o,m_t] -0.245 0.392 -0.885 -0.276 0.609          0.009  2060 1.001 1.004
#> rho[m_o,s]    0.168 0.405 -0.653  0.191 0.829          0.008  2469 1.000 1.001
#> rho[m_t,s]    0.276 0.257 -0.248  0.293 0.725          0.004  3624 1.000 1.002
#>
#> Correlations (posterior mean estimates) in matrix form:
#>        m_o    m_t     s
#> m_o  1.000 -0.245 0.168
#> m_t -0.245  1.000 0.276
#> s    0.168  0.276 1.000
```

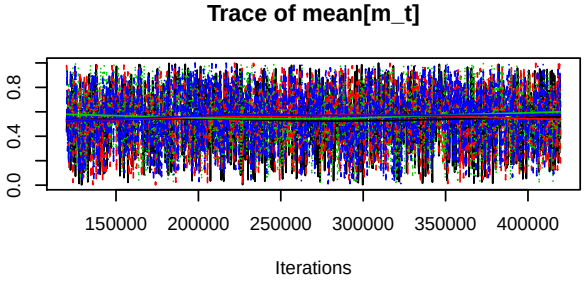
```
#>
#>
#> ##############
#>  Model fit statistics (posterior predictive p-values):
#> Use PPP(fittedModel) to get T1 and T2 posterior predictive checks.
#>
#> Effects of factors on latent scale (additive shift from overall mean):
#>                     Mean    SD   2.5%    50%  97.5% Time-series SE n.eff  Rhat
#> factor_m_o_age[1]  0.037 0.589 -1.082  0.012  1.288          0.010  3563 1.001
#> factor_m_o_age[2] -0.448 0.588 -1.642 -0.447  0.763          0.010  3618 1.001
#> factor_m_o_age[3]  0.411 0.596 -0.643  0.370  1.713          0.010  3775 1.001
#> factor_m_t_age[1] -1.321 1.701 -5.647 -0.982  0.986          0.027  3854 1.006
#> factor_m_t_age[2] -0.523 1.355 -3.683 -0.389  1.929          0.021  4000 1.004
#> factor_m_t_age[3]  1.844 1.892 -0.561  1.421  6.510          0.032  3490 1.007
#> factor_s_age[1]_4 -0.661 0.368 -1.498 -0.623 -0.031          0.006  3601 1.001
#> factor_s_age[2]_5  0.040 0.349 -0.662  0.043  0.752          0.005  4550 1.001
#> factor_s_age[3]_6  0.621 0.360  0.023  0.584  1.404          0.006  4078 1.000
#>                    R_95%
#> factor_m_o_age[1] 1.001
#> factor_m_o_age[2] 1.004
#> factor_m_o_age[3] 1.003
#> factor_m_t_age[1] 1.009
#> factor_m_t_age[2] 1.007
#> factor_m_t_age[3] 1.014
#> factor_s_age[1]_4 1.005
#> factor_s_age[2]_5 1.004
#> factor_s_age[3]_6 1.003
#>
#> Factor SD on latent scale:
#>                      Mean    SD  2.5%   50% 97.5% Time-series SE n.eff  Rhat
#> SD_factor_m_o_age 1.207 1.049 0.410 0.918 3.637          0.017  3890 1.016
#> SD_factor_m_t_age 2.493 2.860 0.476 1.673 9.566          0.049  3345 1.018
#> SD_factor_s_age   1.177 1.071 0.423 0.926 3.319          0.017  3910 1.035
#>                   R_95%
#> SD_factor_m_o_age 1.018
#> SD_factor_m_t_age 1.022
#> SD_factor_s_age   1.036
```

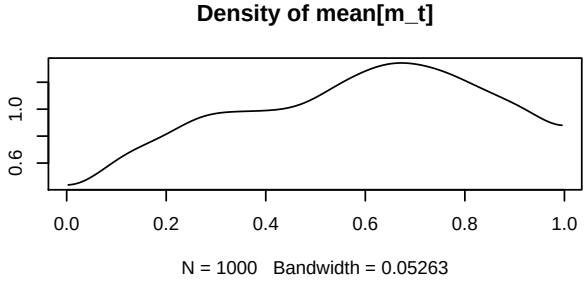
We also check the trace plots for the (group-level) mean parameters:

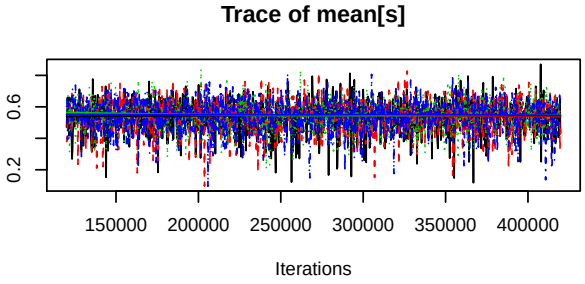
```
plot(fit_bayes_all, parameter = "mean")
```

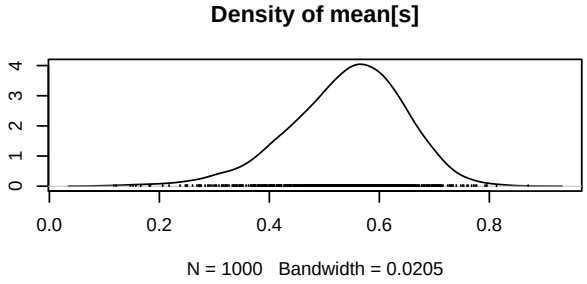






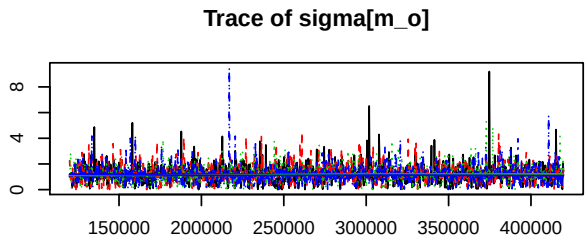


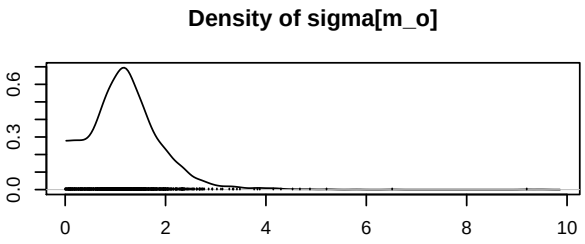


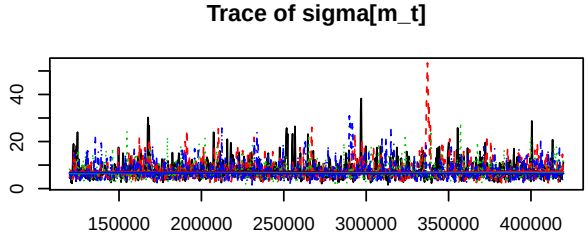


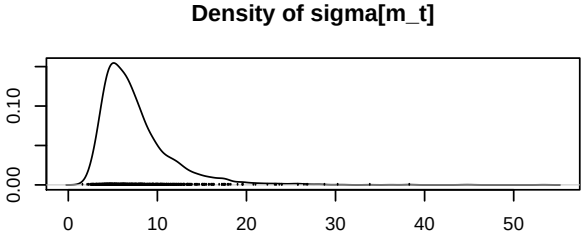
The group-level SD parameters (on probit scale):

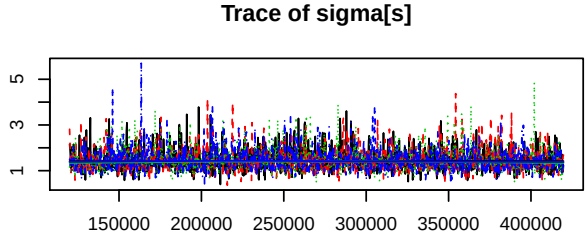
```
plot(fit_bayes_all, parameter = "sigma")
```

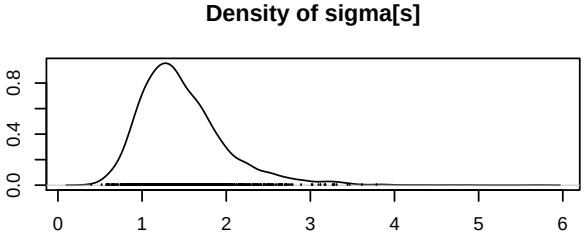






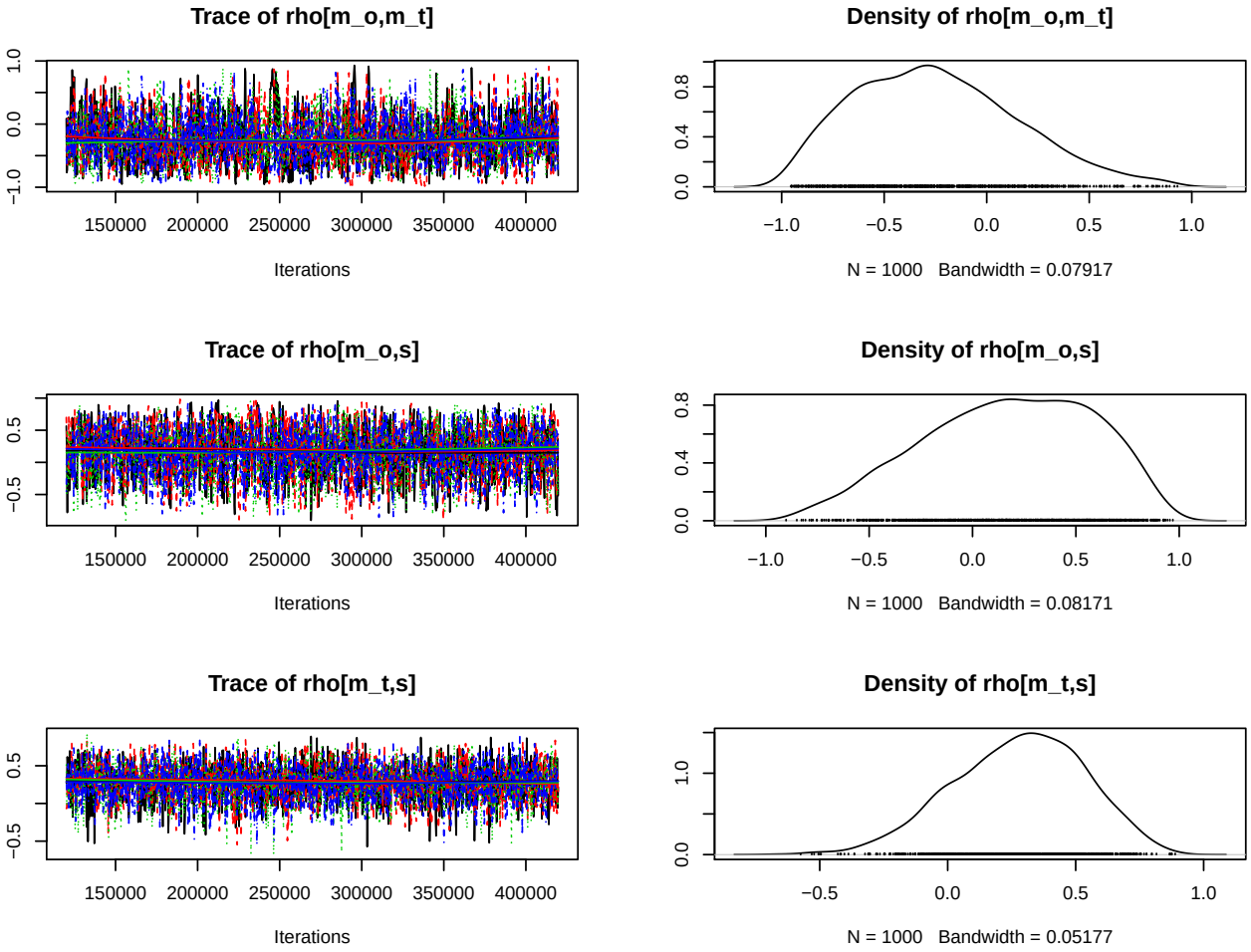






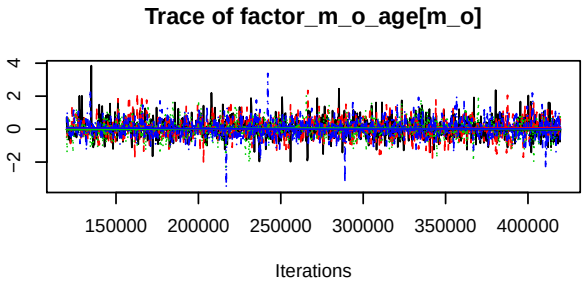
The correlation parameters among the individual effects:

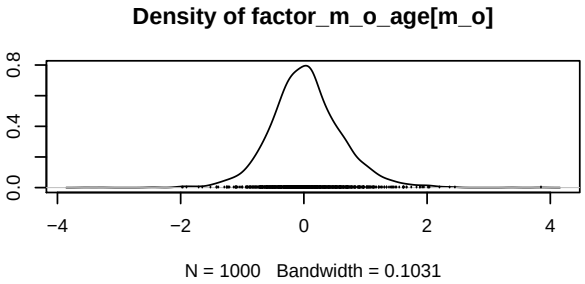
```
plot(fit_bayes_all, parameter = "rho")
```

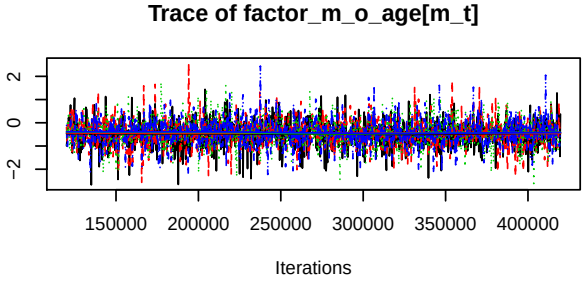


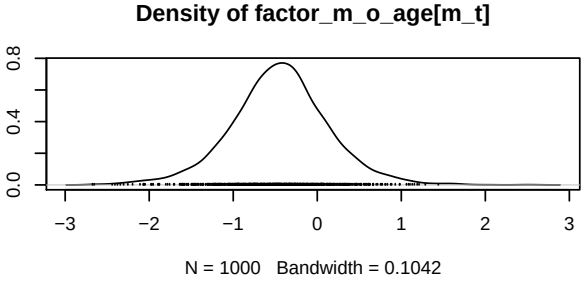
And finally the parameters for the effect of age on the model parameter (as well as their SDs).

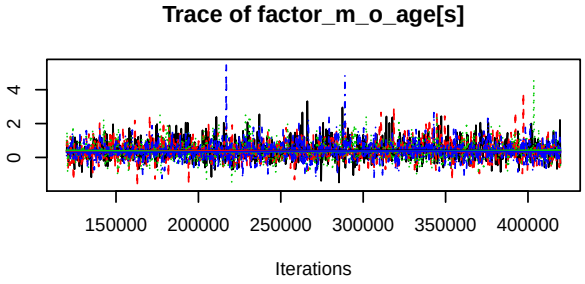
```
plot(fit_bayes_all, parameter = "factor")
```

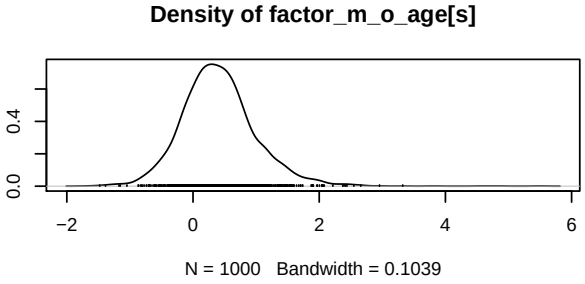

Trace of factor_m_o_age[m_o]
−2 0 2 4
150000 200000 250000 300000 350000 400000
Iterations

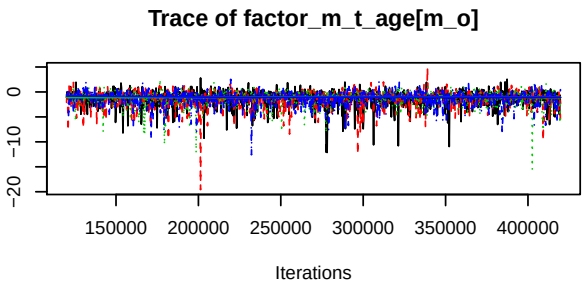

Density of factor_m_o_age[m_o]
0.0 0.4 0.8
−4 −2 0 2 4
N = 1000 Bandwidth = 0.1031

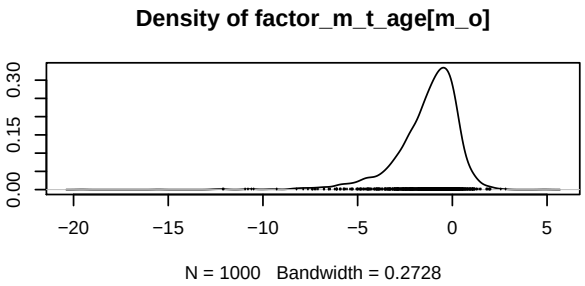

Trace of factor_m_o_age[m_t]
−2 0 2
150000 200000 250000 300000 350000 400000
Iterations

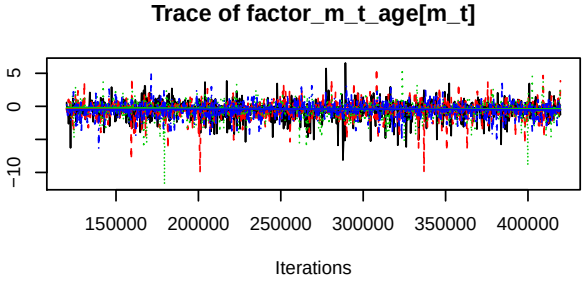

Density of factor_m_o_age[m_t]
0.0 0.4 0.8
−3 −2 −1 0 1 2 3
N = 1000 Bandwidth = 0.1042

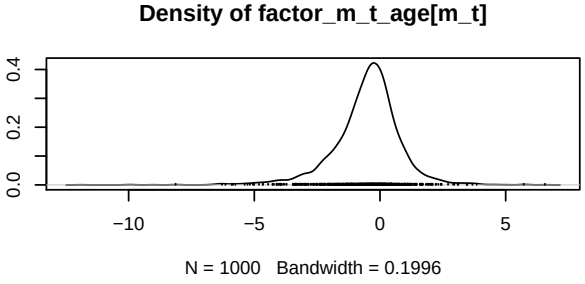

Trace of factor_m_o_age[s]
0 2 4
150000 200000 250000 300000 350000 400000
Iterations

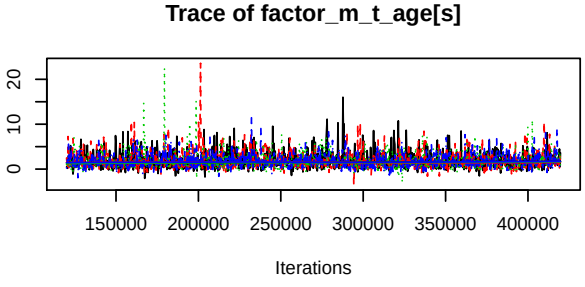

Density of factor_m_o_age[s]
0.0 0.4
−2 0 2 4 6
N = 1000 Bandwidth = 0.1039

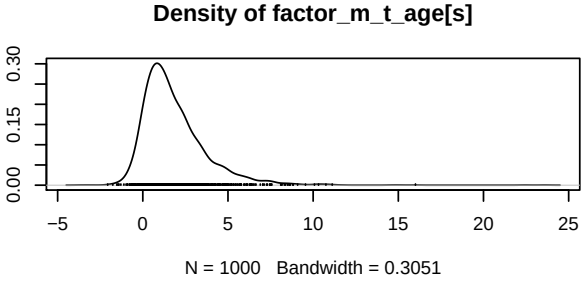

Trace of factor_m_t_age[m_o]
0
−10
−20
150000
200000
250000
300000
350000
400000
Iterations

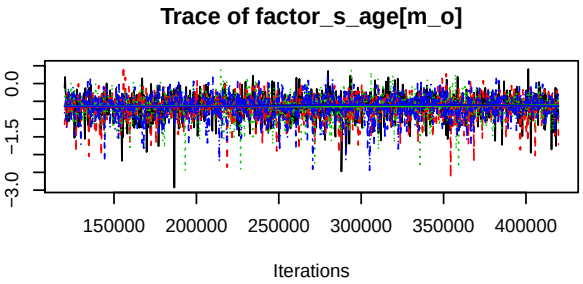

Density of factor_m_t_age[m_o]
0.30
0.15
0.00
−20
−15
−10
−5
0
5
N = 1000 Bandwidth = 0.2728

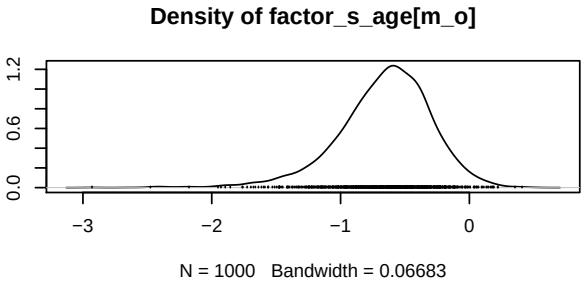

Trace of factor_m_t_age[m_t]
5
0
−10
150000
200000
250000
300000
350000
400000
Iterations

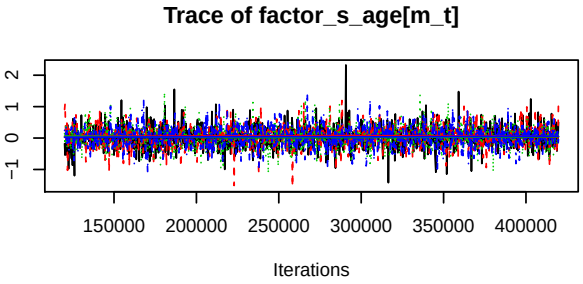

Density of factor_m_t_age[m_t]
0.4
0.2
0.0
−10
−5
0
5
N = 1000 Bandwidth = 0.1996

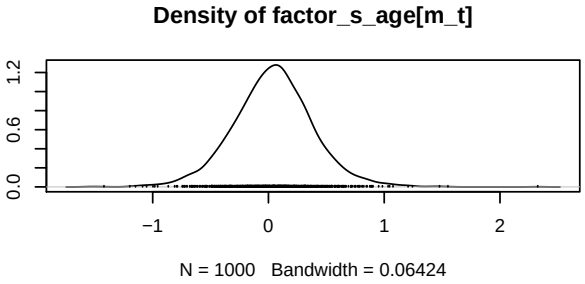

Trace of factor_m_t_age[s]
20
10
0
150000
200000
250000
300000
350000
400000
Iterations

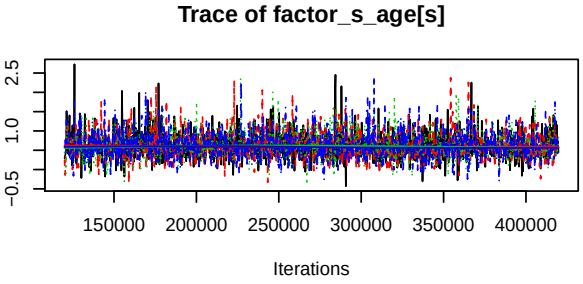

Density of factor_m_t_age[s]
0.30
0.15
0.00
−5
0
5
10
15
20
25
N = 1000 Bandwidth = 0.3051

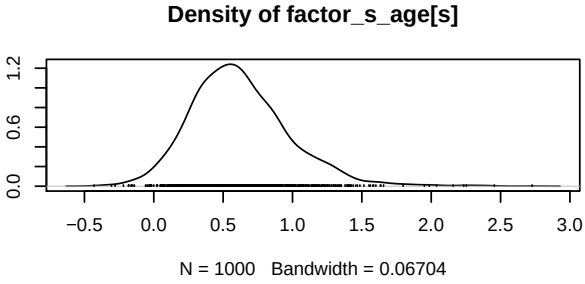

Trace of factor_s_age[m_o]
−3.0
−1.5
0.0
150000
200000
250000
300000
350000
400000
Iterations

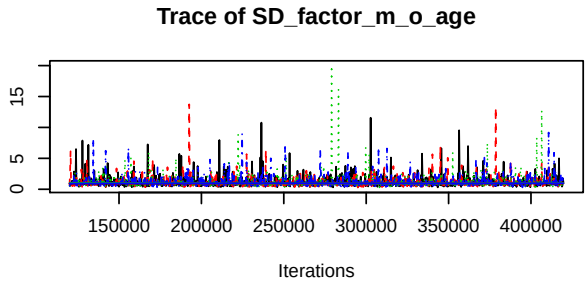

Density of factor_s_age[m_o]
0.0
0.6
1.2
−3
−2
−1
0
N = 1000 Bandwidth = 0.06683

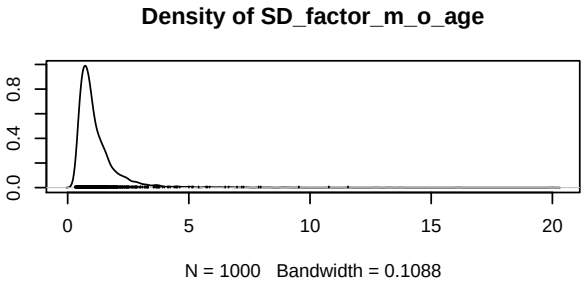

Trace of factor_s_age[m_t]
−1
0
1
2
150000
200000
250000
300000
350000
400000
Iterations

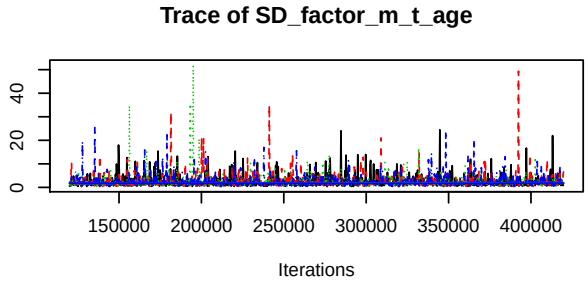

Density of factor_s_age[m_t]
0.0
0.6
1.2
−1
0
1
2
N = 1000 Bandwidth = 0.06424

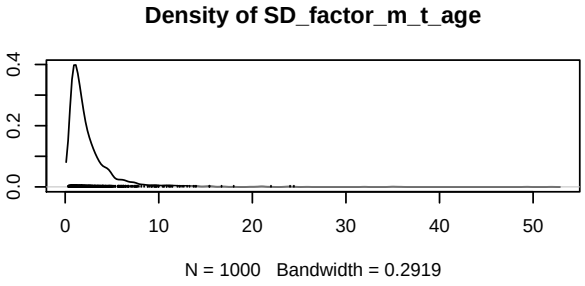

Trace of factor_s_age[s]
−0.5
1.0
2.5
150000
200000
250000
300000
350000
400000
Iterations

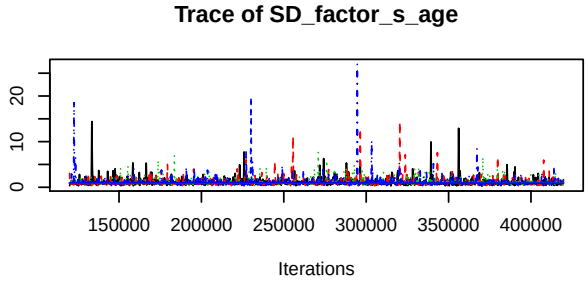

Density of factor_s_age[s]
0.0
0.6
1.2
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
N = 1000 Bandwidth = 0.06704

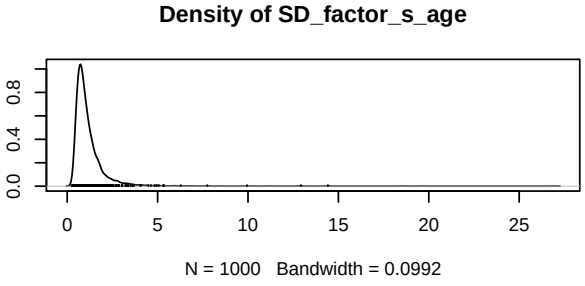












Model Fit We evaluate model fit using posterior predictive tests. There is no evidence for misfit. All posterior predictive p-values are larger than .05. This suggests that the model is able to adequately describe the data.

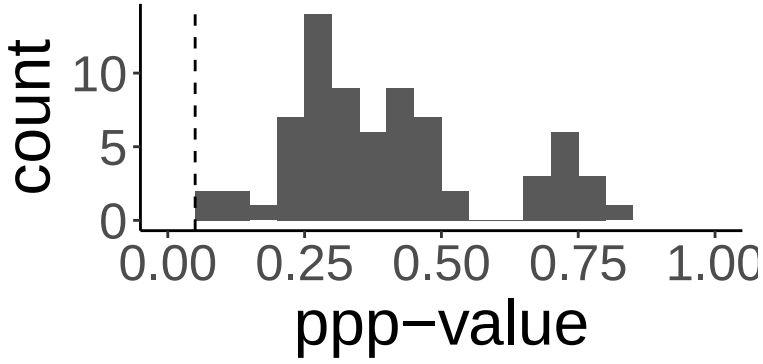
```
ppps
```

```
#>  ## Mean structure (T1):
#>  Observed =  0.07636708 ; Predicted =  0.03366389 ; p-value =  0.145
#>
#> ## Covariance structure (T2):
#>  Observed =  0.5087063 ; Predicted =  0.2788794 ; p-value =  0.139
#>
#> ## Individual fit (T1):
#>     1     2     3     4     5     6     7     8     9    10    11    12    13
#> 0.426 0.743 0.752 0.423 0.299 0.695 0.075 0.743 0.292 0.327 0.712 0.386 0.287
#>    14    15    16    17    18    19    20    21    22    23    24    25    26
#> 0.414 0.295 0.255 0.281 0.120 0.237 0.430 0.245 0.453 0.424 0.290 0.334 0.253
#>    27    28    29    30    31    32    33    34    35    36    37    38    39
#> 0.420 0.286 0.264 0.301 0.339 0.164 0.461 0.681 0.726 0.431 0.385 0.116 0.809
#>    40    41    42    43    44    45    46    47    48    49    50    51    52
#> 0.547 0.698 0.303 0.281 0.787 0.451 0.279 0.286 0.371 0.338 0.759 0.247 0.731
#>    53    54    55    56    57    58    59    60    61    62    63    64    65
#> 0.394 0.309 0.382 0.723 0.471 0.371 0.244 0.228 0.495 0.256 0.219 0.238 0.076
#>    66    67    68    69    70    71    72
```

```
#> 0.527 0.402 0.311 0.466 0.304 0.492 0.402
```

This is also visible when plotting the distribution of individual *ppp*-values.

```
ggplot(data = as_tibble(ppps$ind.T1.p),
       mapping = aes(value)) +
  geom_histogram(binwidth = 0.05, boundary = 0) +
  coord_cartesian(xlim = c(0, 1)) +
  geom_vline(xintercept = 0.05, linetype = "dashed") +
  xlab("ppp-value")
```



Group-Level Estimates We can investigate the group-level parameter estimates (peaks) and 80% HDIs in tabular form:

```
mus = gather_draws(fit_bayes_all$runjags, mu[i]) %>%
    mutate(
      parameter = factor(
        i,
        levels = c(3, 1, 2),
        labels = c("italic(s)", "italic(m[o])", "italic(m[t])"))
    ) %>%
    rename(mean = .value) %>%
    ungroup() %>%
    select(-i, -.variable)

pars = gather_draws(fit_bayes_all$runjags,
                    `factor_.*_age`[i], regex = TRUE) %>%
  ungroup() %>%
  mutate(age_group = factor(age_groups[i], level = age_groups),
         #estimate = pnorm(.value),
         parameter = factor(
           str_replace(str_replace(.variable, "factor_", ""), "_age", ""),
           levels = c("s", "m_o", "m_t"),
           labels = c("italic(s)", "italic(m[o])", "italic(m[t])"))) %>%
  mutate(age_group2 = factor(age_group, levels = rev(levels(di$age))))
pars = left_join(pars, mus) %>%
    mutate(estimate = pnorm(.value + mean))
```

```
#> Joining, by = c(".chain", ".iteration", ".draw", "parameter")
```

```
pars %>%
  group_by(parameter, age_group) %>%
  mode_hdci(estimate, .width = c(0.80)) %>%
  mutate(.lower = round(.lower, 4))
```

```
#> # A tibble: 9 x 8
#> # Groups:   parameter [3]
#>   parameter    age_group estimate .lower .upper .width .point .interval
#>   <fct>        <fct>        <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
#> 1 italic(s)    4           0.304   0.12   0.491    0.8 mode   hdci
#> 2 italic(s)    5           0.595   0.365  0.794    0.8 mode   hdci
#> 3 italic(s)    6           0.786   0.634  0.914    0.8 mode   hdci
#> 4 italic(m[o]) 4           0.997   0.961  1.00     0.8 mode   hdci
#> 5 italic(m[o]) 5           0.993   0.904  1.00     0.8 mode   hdci
#> 6 italic(m[o]) 6           0.999   0.984  1        0.8 mode   hdci
#> 7 italic(m[t]) 4           0.0227  0      0.635    0.8 mode   hdci
#> 8 italic(m[t]) 5           0.0376  0      0.804    0.8 mode   hdci
#> 9 italic(m[t]) 6           0.991   0.702  1        0.8 mode   hdci
```

To investigate the difference between age groups, we calculate the difference distribution of the group-level posterior for each pairwise comparison of age groups for each parameter. We first look at the 80% and 95% highest density intervals of the difference distributions.

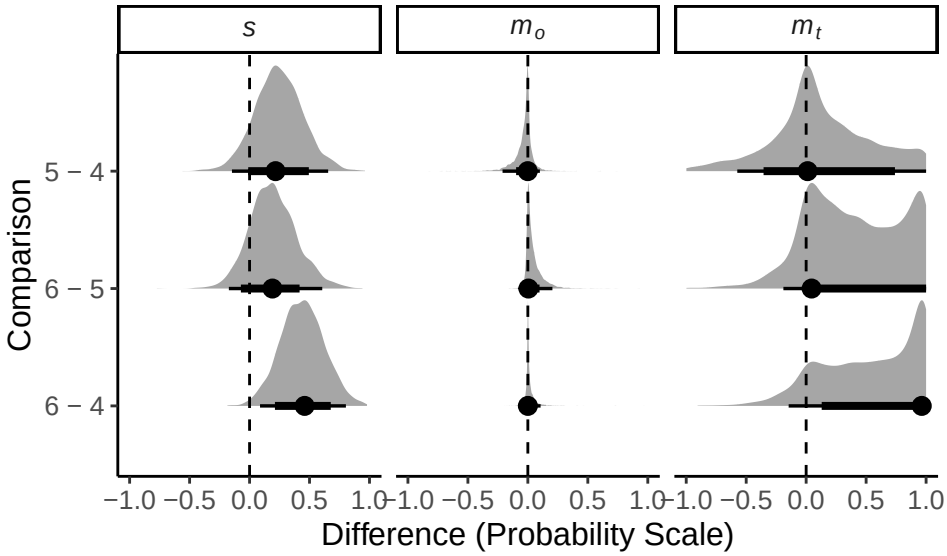
```
tmp_diff = pars %>%
  mutate(parameter = factor(parameter, levels =
                              c("italic(s)", "italic(m[o])", "italic(m[t])"),
                            labels = c("s", "m_o", "m_t"))) %>%
  mutate(inter = parameter:age_group) %>%
  compare_levels(estimate, by = inter) %>%
  mutate(par1 = str_extract(as.character(inter), "[a-z_]+"),
         par2 = str_extract(substr(as.character(inter), 5, 100),
                            "[a-z_]+")) %>%
  filter(par1 == par2) %>%
  droplevels %>% ungroup %>%
  mutate(diff = str_remove_all(as.character(inter), "[a-z_:]+")) %>%
  mutate(inter =
           factor(inter,
                  levels = c("m_o:6 - m_o:4", "m_o:6 - m_o:5", "m_o:5 - m_o:4",
                             "m_t:6 - m_t:4", "m_t:6 - m_t:5", "m_t:5 - m_t:4",
                             "s:6 - s:4", "s:6 - s:5", "s:5 - s:4"))) %>%
  mutate(diff =
           factor(diff, levels = c("6 - 4", "6 - 5", "5 - 4"))) %>%
  mutate(parameter = factor(par1, levels = c("s", "m_o", "m_t"),
                            labels = c("italic(s)",
                                       "italic(m[o])",
                                       "italic(m[t])")))

tmp_diff %>%
  group_by(inter) %>%
  mean_hdci(estimate, .width = c(0.80, 0.95))
```

```
#> # A tibble: 18 x 7
#>    inter         estimate  .lower .upper .width .point .interval
#>    <fct>            <dbl>   <dbl>  <dbl>  <dbl> <chr>  <chr>
#>  1 m_o:6 - m_o:4   0.0150 -0.0197 0.0444   0.8  mean   hdci
#>  2 m_o:6 - m_o:5   0.0465 -0.0217 0.0972   0.8  mean   hdci
#>  3 m_o:5 - m_o:4  -0.0315 -0.0979 0.0461   0.8  mean   hdci
#>  4 m_t:6 - m_t:4   0.537   0.130  1        0.8  mean   hdci
#>  5 m_t:6 - m_t:5   0.406   0.0333 1        0.8  mean   hdci
#>  6 m_t:5 - m_t:4   0.131  -0.355  0.740    0.8  mean   hdci
```

```
#>  7 s:6 - s:4       0.440   0.212  0.677    0.8  mean   hdci
#>  8 s:6 - s:5       0.196  -0.0725 0.417    0.8  mean   hdci
#>  9 s:5 - s:4       0.243  -0.0114 0.494    0.8  mean   hdci
#> 10 m_o:6 - m_o:4   0.0150 -0.0565 0.106    0.95 mean   hdci
#> 11 m_o:6 - m_o:5   0.0465 -0.0506 0.205    0.95 mean   hdci
#> 12 m_o:5 - m_o:4  -0.0315 -0.211  0.100    0.95 mean   hdci
#> 13 m_t:6 - m_t:4   0.537  -0.146  1        0.95 mean   hdci
#> 14 m_t:6 - m_t:5   0.406  -0.190  1        0.95 mean   hdci
#> 15 m_t:5 - m_t:4   0.131  -0.574  1.00     0.95 mean   hdci
#> 16 s:6 - s:4       0.440   0.0872 0.803    0.95 mean   hdci
#> 17 s:6 - s:5       0.196  -0.173  0.606    0.95 mean   hdci
#> 18 s:5 - s:4       0.243  -0.147  0.655    0.95 mean   hdci
```

We can also plot the difference distributions plus 80% and 95% credibility intervals. From this it is clear that there is not really any evidence for differences in m_o. However, there is some evidence for a difference in m_t. Furthermore, the monotonic increase for$s$ also receives some support.

```
tmp_diff %>%
  ggplot(aes(y = diff, x = estimate)) +
  stat_halfeye(.width = c(0.80, 0.95), point_interval = mode_hdci,
               normalize = "groups") +
  geom_vline(xintercept = 0, linetype = "dashed") +
  facet_wrap(~parameter, labeller = label_parsed) +
  xlab("Difference (Probability Scale)") +
  ylab("Comparison") +
  theme(text = element_text(size = 12))
```

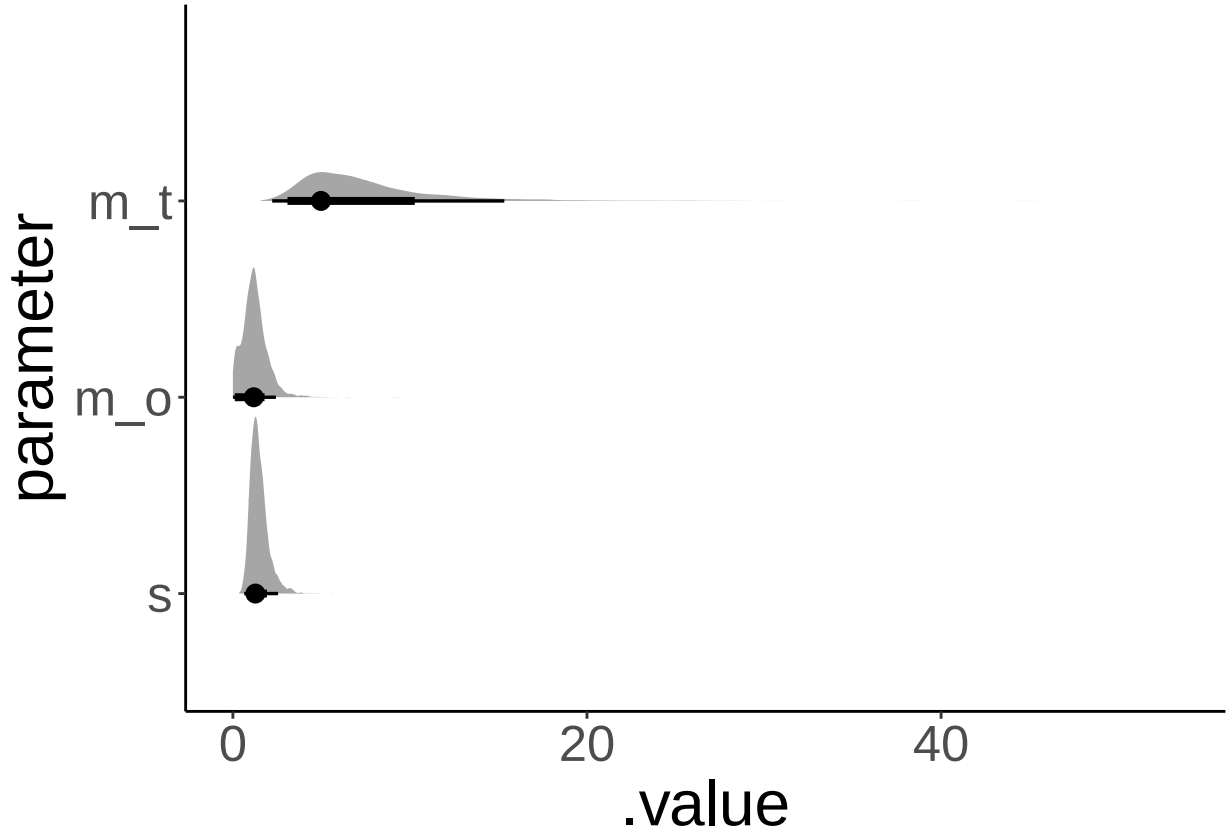


Individual-Level Parameters and Individual Differences We first inspect the posterior distribution of the sigma parameters.

```
sigmas = gather_draws(fit_bayes_all$runjags, sigma[par]) %>%
   mutate(parameter = factor(par,
                             levels = c(3, 1, 2),
                             labels = c("s", "m_o", "m_t")))
sigmas %>%
  ggplot(aes(y = parameter, x = .value)) +
  stat_halfeye(.width = c(0.80, 0.95), point_interval = mode_hdci)
```

```
sigmas %>%
  group_by(parameter) %>%
  mode_hdci(.value, .width = c(0.80, 0.95))
```

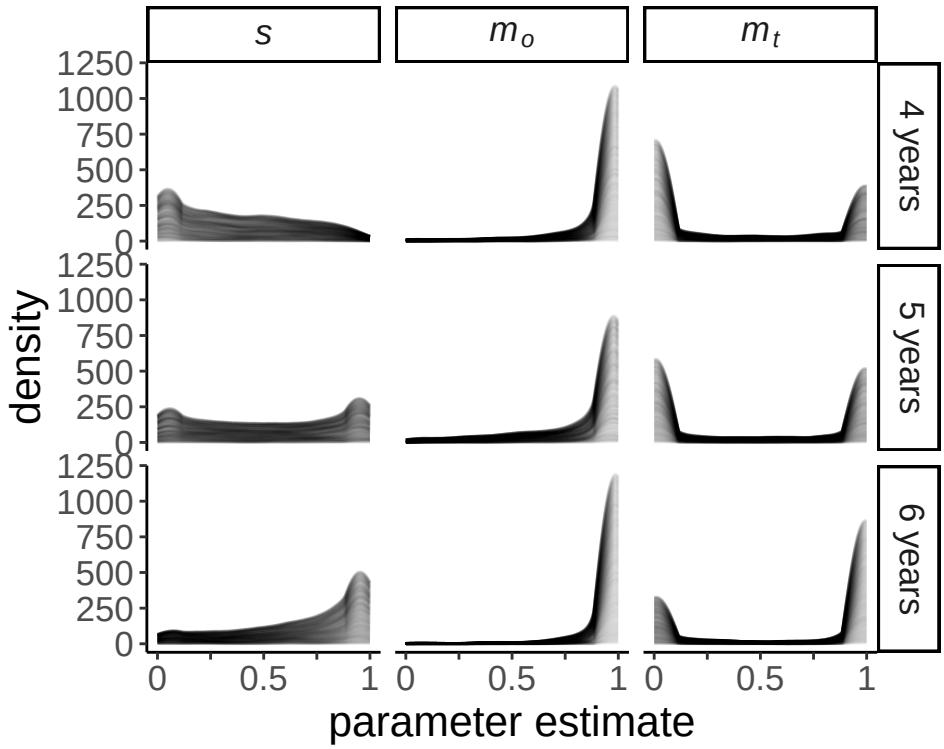
```
#> # A tibble: 6 x 7
#>   parameter .value  .lower .upper .width .point .interval
#>   <fct>      <dbl>   <dbl>  <dbl>  <dbl> <chr>  <chr>
#> 1 s           1.27 0.823     1.92   0.8  mode   hdci
#> 2 m_o         1.18 0.106     1.79   0.8  mode   hdci
#> 3 m_t         4.97 3.08     10.3    0.8  mode   hdci
#> 4 s           1.27 0.637     2.55   0.95 mode   hdci
#> 5 m_o         1.18 0.00296   2.43   0.95 mode   hdci
#> 6 m_t         4.97 2.22     15.3    0.95 mode   hdci
```




```
samp1 = gather_draws(fit_bayes_all$runjags, theta[i,participant2])

samp1 = di %>%
  mutate(participant2 = 1:nrow(di)) %>%
  select(participant2, age_group, age2) %>%
  right_join(samp1, by = "participant2") %>%
  mutate(parameter =
           factor(i, levels = c(3, 1, 2),
                  labels = c("italic(s)", "italic(m[o])", "italic(m[t])"))) %>%
  mutate(age_group2 = factor(age_group,
                             levels = c("4", "5", "6"),
                             labels = c(c("4~years", "5~years", "6~years"))))
```

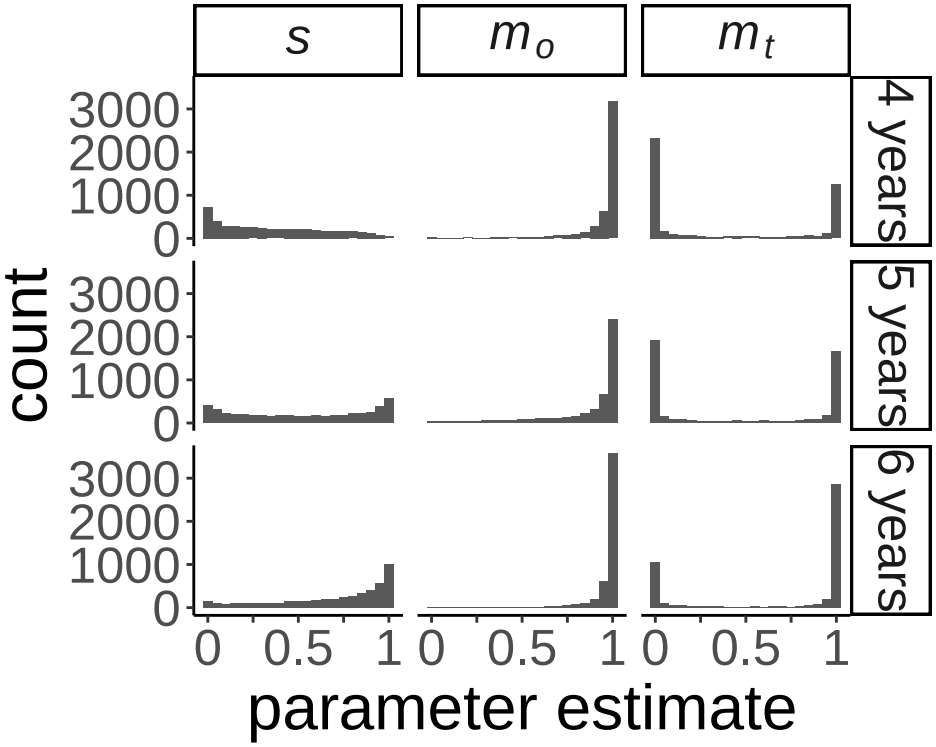
In addition to the group-level parameters, we can also investigate the distribution of individual-level parameters. For this, we randomly sample 200 of the 4000 posterior samples and plot the density of the individual-level parameter distribution separately per parameter and age group. We plot the individual densities using an alpha-value of 0.05 so that darker regions represent regions with overlap and therefore higher density among the distribution of the individual-level parameters.

```
rows = sample.int(max(samp1$.draw), 200)

samp1 %>%
  filter(.draw %in% rows) %>%
  ggplot(aes(x = .value,
             group = .draw,
             fill = .value)) +
  stat_density(geom = "line",
               alpha = 0.05,
               adjust = 1,
               bw = 0.05,
               kernel = "o") +
  facet_grid(age_group2~parameter,
             labeller = label_parsed) +
  scale_x_continuous(breaks = c(0, 0.25, 0.5, 0.75, 1),
                     labels = c("0", "", "0.5", "", "1")) +
  xlab("parameter estimate") +
  theme(text = element_text(size = 16))
```



To ensure this pattern is not an artifact of applying a density estimator individually to each posterior sample, we can simply plot a histogram for these data, which ignore the individual posterior draws. Should this show a very different pattern, this should provide some caution towards the density estimators. However, the pattern looks pretty much the same as when using the density plot.

```r
samp1 %>%
  filter(.draw %in% rows) %>%
  ggplot(aes(x = .value, fill = .value)) +
  geom_histogram(binwidth = 0.05) +
  facet_grid(age_group2 ~ parameter,
             labeller = label_parsed) +
  scale_x_continuous(breaks = c(0, 0.25, 0.5, 0.75, 1),
                     labels = c("0", "", "0.5", "", "1")) +
  xlab("parameter estimate")
```



For s, we see that the distributiuons of individual-level parameters are a lot less peaked than for the other two parameters. For the 4-year olds we see a small peak around 0 with the remaining probability mass distributed more towards the left side of the parameter space. For the 5-year olds we see a slightly bimodal pattern, but also some evidence for an almost uniform distribution. For the 6-year olds we see almost a mirror pattern of the 4-year olds with a larger peak at 1 and some evidence for a small peak at 0.

For m_o we can see that the vast majority of individual-level estimates is huddled towards the right boundary of the parameter space. Very few individual-level parameter estimates appear to be below .75. Furthermore, there seem to be few differences between conditions, which is consistent with the pattern of the group-level parameters.

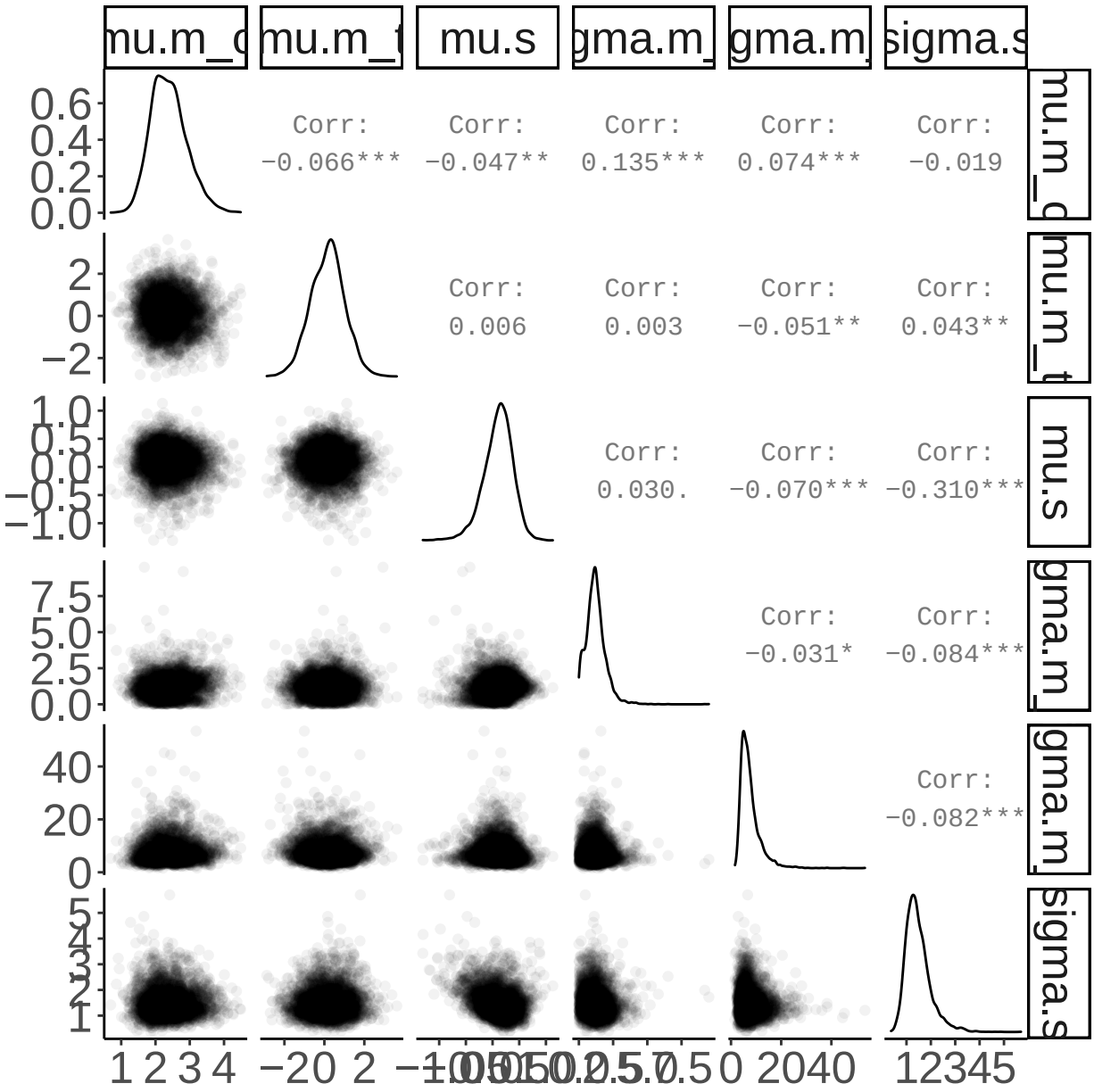
For m_t we see quite a different pattern suggesting a bimodal distribution with the modes being at either ends of the parameter space. Furthermore, the weight given to the two modes seem to shift across age-groups with the 4 and 5-year olds having a larger peak at 0, whereas for the 6-year olds there appears to be a larger peak at 1. This bimodal pattern is consistent with the large SD of the group-level distribution (posterior mean = 7.56), which is also considerably larger than these SDs for the other two parameters (posterior means < 1.5). The reason for this bimodality given a normal group-level distribution is that the latent-trait model assumes a normal distribution on an unconstrained latent space, which is then transformed into the probability space using the probit transformation.

When comparing the individual-level distributions with the group-level distributions, we see quite a few

differences. This suggests that the group-level mean does not necessarily provide a good representation of all individual parameters. One posible reason is parameter trade-offs at the group-level. We investigated this with the next plot.

```
gather_draws(fit_bayes_all$runjags, mu[par], sigma[par]) %>%
  mutate(parameter = factor(par, levels = 1:3,
                            labels = c("m_o", "m_t", "s")),
    inter = paste0(.variable, ".", parameter)) %>%
  ungroup() %>%
  select(.draw, inter, .value) %>%
  spread("inter", ".value") %>%
  select(-.draw) %>%
  GGally::ggpairs(progress = FALSE,
                  lower = list(continuous = GGally::wrap("points",
                                                         alpha = 0.05)))
```

```
#> Registered S3 method overwritten by 'GGally':
#>   method from
#>   +.gg   ggplot2
```



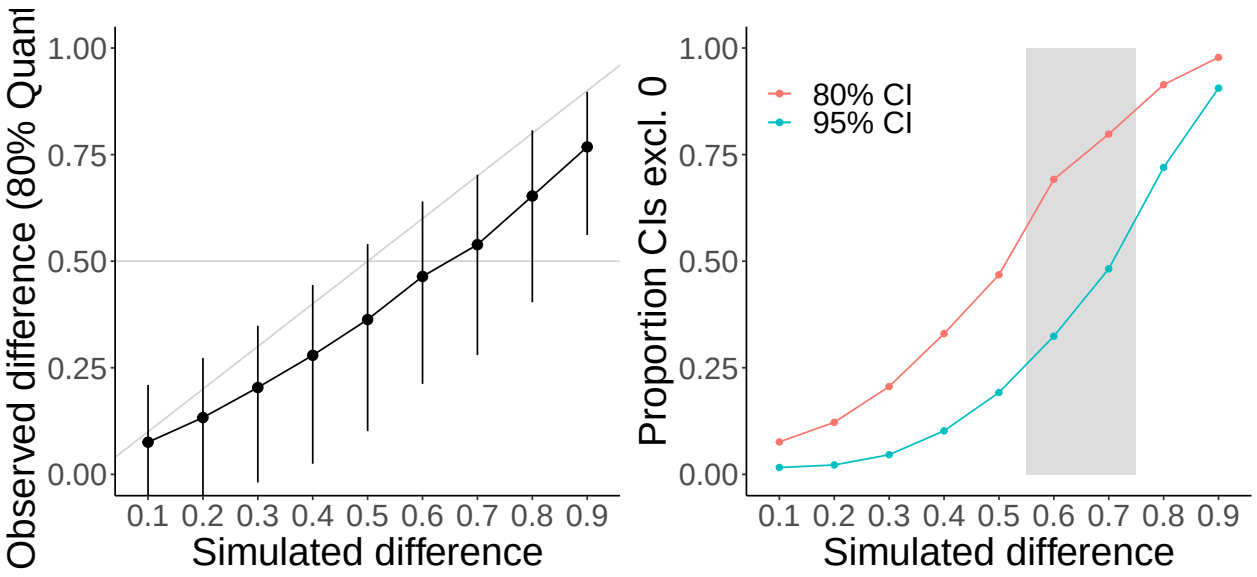
This plot provides little evidence for large parameter trade-offs. We see some mild positive correlation, $r = .135$, between μ and σ for m_o. Furthermore, a negative correlation of $r = -.31$ between μ and σ of s. Overall this suggests that parameter trade-offs cannot be mainly responsible for the disagreement between individual-level and group-level.

Bayesian Power Analysis We performed a simulation study that can be seen as the Bayesian equivalent of a power analysis. The goal was to see what type of difference in the s parameter could be reliably detected using the 80% and 95% credibility interval with the other parameter estimates fixed to values as obtained from our study. One problem with this analysis is that it was a retroactive analysis, that is, performed after obtaining the data and thus presupposes that our mean estimates of the other parameter are precise enough. As just described, with the exception being the s parameter which was investigated in this simulation, all other parameters were fixed to values similar to the one obtained in our experiment.

In particular, we performed a simulation in which we simulated data from two groups that were in size identical to the groups in our study (i.e., same number of participants and same number of trials as in each of our age groups) and then refit the data with the same MPT model as in the main study (but for two

groups only). In the simulation, we systematically varied the difference in the s parameter from 0.1 to 0.9 (and mean $s = 0.5$) with all other parameters fixed to the same values. For each value we simulated 500 data sets. The full details of the simulation can be found in the `R` files `simulation.R` (which contains the code that runs all simulations) and `simfun.R` (which contains the function that performs one simulation and is called repeatedly in `simulation.R`). The results are shown in the following figure (description below).

```
load("cache/sim_results.rda")
sim_overview = bind_rows(allfit) %>%
  group_by(sim_diff) %>%
  summarise(across(c(starts_with("ci"), obs_diff),
                   .fns = list(mean = mean, sd = sd,
                               lower = ~quantile(., probs = 0.1),
                               upper = ~quantile(., probs = 0.8))))

p1 = ggplot(sim_overview, aes(x = factor(sim_diff))) +
  geom_abline(slope = 0.1, intercept = 0, color = "lightgrey") +
  geom_hline(yintercept = 0.5, colour = "lightgrey") +
  geom_line(aes(y = obs_diff_mean, group = 1)) +
  geom_pointrange(aes(y = obs_diff_mean,
                      ymin = obs_diff_lower,
                      ymax = obs_diff_upper)) +
  coord_cartesian(ylim = c(0, 1)) +
  labs(x = "Simulated difference",
       y = "Observed difference (80% Quantile)")

p2 = ggplot(sim_overview, aes(x = factor(sim_diff))) +
  scale_x_discrete() +
  annotate("rect", xmin = 5.5, xmax = 7.5, ymin = 0, ymax = 1, alpha = 0.2) +
  geom_line(aes(y = ci80_mean, colour = "80% CI", group = 1)) +
  geom_point(aes(y = ci80_mean, colour = "80% CI")) +
  geom_line(aes(y = ci95_mean, colour = "95% CI", group = 1)) +
  geom_point(aes(y = ci95_mean, colour = "95% CI")) +
  labs(x = "Simulated difference",
       y = "Proportion CIs excl. 0") +
  coord_cartesian(ylim = c(0, 1)) + theme(
    legend.position = c(.35, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6),
    legend.title = element_blank()
    )
wrap_plots(p1, p2)
#ggsave("calibration.png")
```



One interesting result coming from the simulation is that we see quite a bit of hierarchical or prior-based shrinkage. This result is shown in the left panel. It shows on the x-axis the simulated (i.e., "true" in the simulation) difference between the two groups in the s parameter and on the y-axis the observed difference between the two groups in the s parameter (error bars show th 80% quantile intervals of the simulation). What is clear from the plot is that the observed difference is always smaller than the simulated difference (i.e., the difference is shrunk towards 0). This can be seen by comparing the observed difference against the main diagonal (in gray) showing the identity line; all observed data points are below the line. We have also added a horizontal line at 0.5, which is roughly the difference in the s parameter we have actually observed. We can see that such an observed difference is most likely if the "true" (i.e., simulated) difference is somewhere between 0.6 and 0.7.

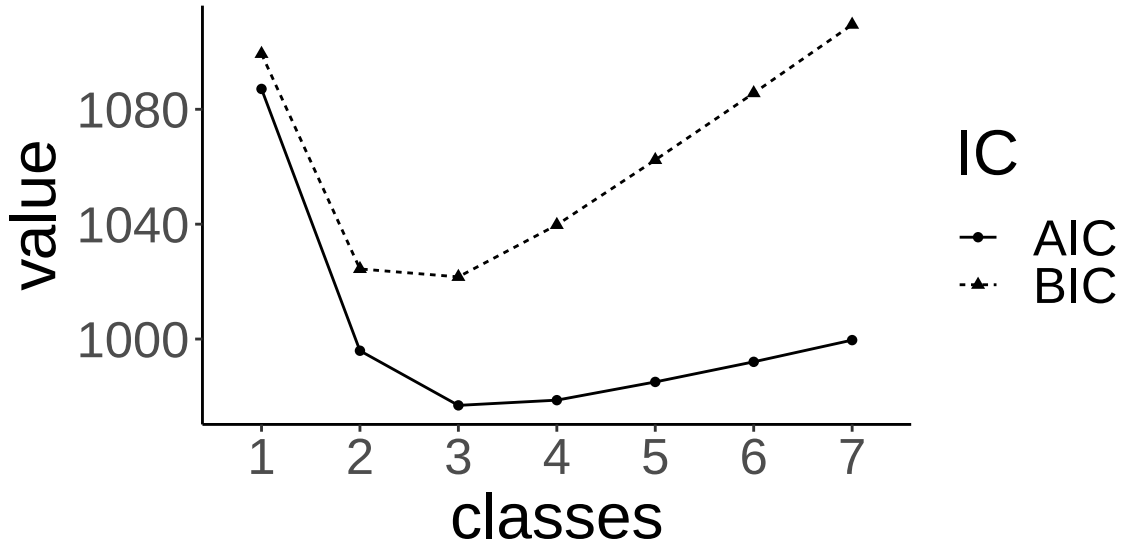
The right panel shows the power of our Bayesian analysis. That is, how often the Bayesian credibility interval (80% or 95%) of the difference distribution for s between the two groups excludes 0 (this is the criterion we use in the paper). We have shaded the area that seems to correspond to our observed difference (i.e., a simulated difference of around 0.6 and 0.7). We can see that the power for the 80% interval in the region that seems to correspond to our observed difference is around 0.75.

Latent Class Approach

To further explore the individual differences we estimated the MPT model using a latent-trait approach. This approach tries to distribute the observed individual participants into a number of prespecified classes. To identify the number of classes that are supported by the data, we can use both AIC and BIC. As shown below, both AIC and BIC suggest three classes (for AIC, the second best number of classes is 2, whereas for BIC it is 4).

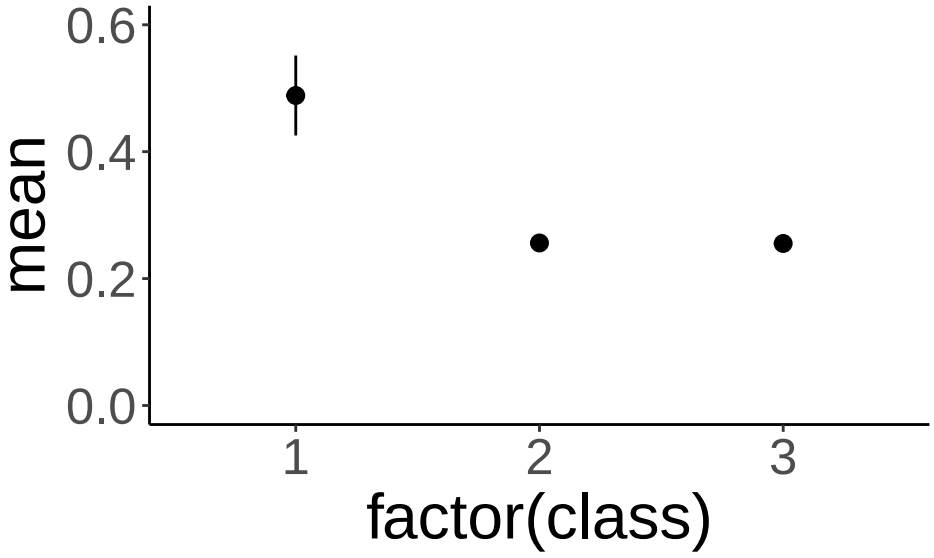
```
hmm_all = read_lines("models/hmm_all.txt")
hmm_model_sel = tibble(classes = factor(1:7),
  AIC = as.numeric(str_extract(hmm_all[str_starts(hmm_all, pattern = "AIC:")],
                               "\\d+.\\d+")),
  BIC = as.numeric(str_extract(hmm_all[str_starts(hmm_all, pattern = "BIC:")],
                               "\\d+.\\d+")))

hmm_model_sel %>%
  gather("IC", "value", -classes) %>%
  ggplot(aes(x = classes, y = value,
             group = IC, shape = IC, linetype = IC)) +
  geom_line() +
```

```
  geom_point()
```



The following shows the class weights for the 3 classes with 95% CIs.

```
class_weights = read.table(text = "class mean x1 lower upper x2
    1            0.488536  [ 0.425518 0.551554 ]
    2            0.256105  [ 0.256105 0.256105 ]
    3            0.255359  [ 0.255359 0.255359 ] ",
    header = TRUE) %>%
  select(-x1, -x2)

ggplot(data = class_weights,
       mapping = aes(x = factor(class),
                     y = mean,
                     ymin = lower,
                     ymax = upper)) +
  geom_pointrange() +
  coord_cartesian(ylim = c(0, 0.6))
```



```
prob_all_3 = read.table(text = "Class1  Class2  Class3
    1  0.778342  0.221118  0.000539
    2  0.933240  0.066713  0.000047
    3  0.993203  0.005675  0.001122
    4  0.111495  0.000016  0.888488
    5  0.995100  0.004091  0.000809
    6  0.988116  0.011810  0.000074
    7  0.608450  0.391549  0.000001
    8  0.909574  0.090363  0.000063
    9  0.021995  0.978005  0.000000
   10  0.002698  0.997302  0.000000
   11  0.933240  0.066713  0.000047
   12  0.949402  0.029961  0.020637
   13  0.021995  0.978005  0.000000
   14  0.669720  0.001689  0.328591
   15  0.005985  0.000001  0.994015
   16  0.005985  0.000001  0.994015
   17  0.998627  0.000494  0.000879
   18  0.157501  0.842499  0.000000
   19  0.005985  0.000001  0.994015
   20  0.669720  0.001689  0.328591
   21  0.998627  0.000494  0.000879
   22  0.219580  0.780413  0.000007
   23  0.980709  0.002679  0.016612
   24  0.002698  0.997302  0.000000
   25  0.831118  0.156878  0.012003
   26  0.005985  0.000001  0.994015
   27  0.032741  0.967258  0.000001
   28  0.002698  0.997302  0.000000
   29  0.005985  0.000001  0.994015
   30  0.005985  0.000001  0.994015
   31  0.827949  0.172047  0.000004
   32  0.157501  0.842499  0.000000
   33  0.778342  0.221118  0.000539
   34  0.973390  0.003696  0.022914
   35  0.627250  0.372721  0.000029
   36  0.716448  0.282862  0.000690
   37  0.111495  0.000016  0.888488
   38  0.157501  0.842499  0.000000
   39  0.953707  0.045298  0.000995
   40  0.168368  0.831624  0.000007
   41  0.990579  0.007866  0.001555
   42  0.002698  0.997302  0.000000
   43  0.005985  0.000001  0.994015
   44  0.909574  0.090363  0.000063
   45  0.015925  0.984075  0.000000
   46  0.082817  0.000017  0.917166
   47  0.998093  0.000686  0.001221
   48  0.527891  0.472108  0.000001
   49  0.527891  0.472108  0.000001
   50  0.953707  0.045298  0.000995
   51  0.082817  0.000017  0.917166
   52  0.953707  0.045298  0.000995
```

```
   53  0.700474  0.299502  0.000023
   54  0.745667  0.011250  0.243082
   55  0.669720  0.001689  0.328591
   56  0.993203  0.005675  0.001122
   57  0.983560  0.016337  0.000103
   58  0.119625  0.000144  0.880231
   59  0.005985  0.000001  0.994015
   60  0.005985  0.000001  0.994015
   61  0.652889  0.000198  0.346913
   62  0.082817  0.000017  0.917166
   63  0.005985  0.000001  0.994015
   64  0.005985  0.000001  0.994015
   65  0.997074  0.002769  0.000157
   66  0.652889  0.000198  0.346913
   67  0.778342  0.221118  0.000539
   68  0.002698  0.997302  0.000000
   69  0.219580  0.780413  0.000007
   70  0.981669  0.000323  0.018008
   71  0.219580  0.780413  0.000007
   72  0.980709  0.002679  0.016612")

di3 = bind_cols(di, prob_all_3)

di3 %>%
  select(participant, Class1:Class3) %>%
  gather("class", "prob", -participant) %>%
  group_by(participant) %>%
  summarize(min_prob = max(prob)) %>%
  {psych::describe(.$min_prob)} %>%
  print_table()
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
X1	1	72	0.89	0.13	0.95	0.91	0.06	0.53	1	0.47	-1.12	0.12	0.02

We then add the individuals' posterior probability of class membership to the existing data. Those show a relatively clear pattern, the lowest class membership probability is 0.53 with mean and median at around .9.

```
d3 = di3 %>%
  gather("class", "prob", Class1:Class3) %>%
  group_by(participant, age_group, months, age2) %>%
  summarise(class = class[which.max(prob)],
            prob_sel = max(prob))

with(d3, table(age_group, class))
```

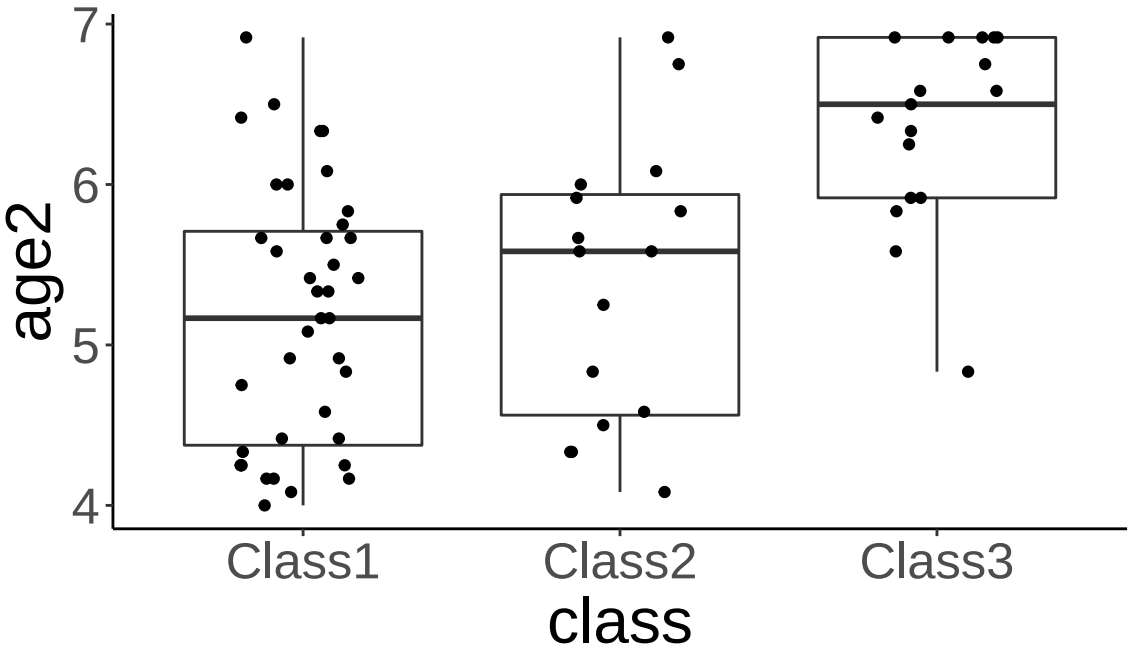
```
#>           class
#> age_group Class1 Class2 Class3
#>         4     17      6      1
#>         5     14      6      4
#>         6      8      4     12
```

If we simply classify participants based on their maximum posterior probability of class membership, we see that the largest class, class 1, has almost equally many members from age 4 and age 5, with a slight majority for age 4. Additionally, still 8 6-year olds are classified into class 1. Class 2 consists of an almost

equal amount of members from all age-groups, with only 6-year olds, slightly under-represented. Class 3 finally laregly consists of 6-year olds.

The following figure shows the age in month as a function of class membership. This figure clearly shows that age increases monotonically when going from class 1 to class 3.

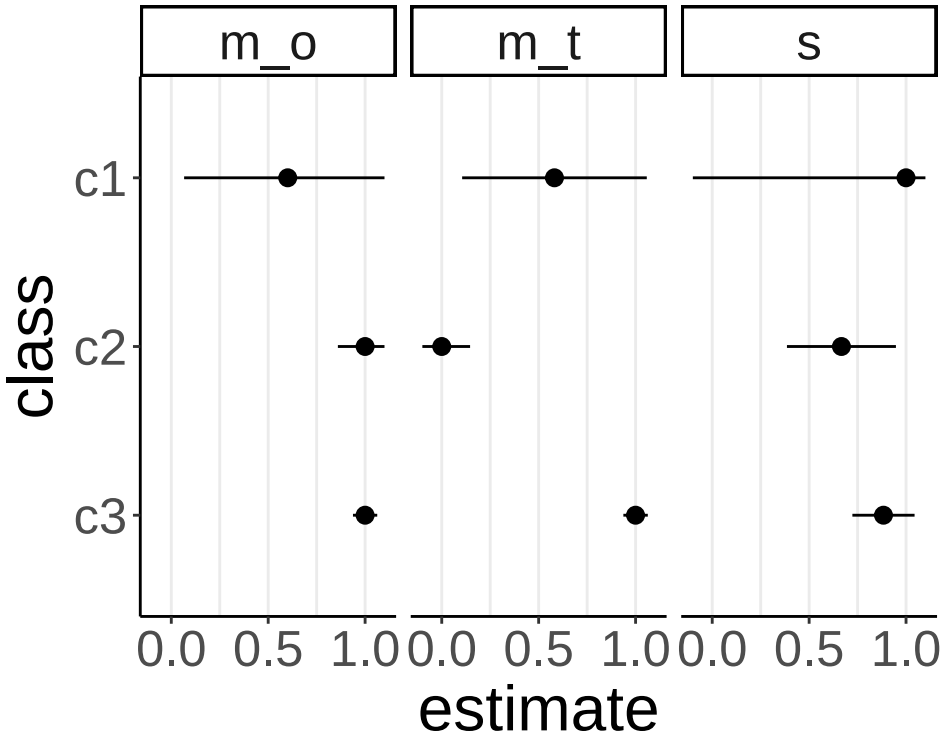
```
ggplot(d3, aes(x = class, y = age2)) +
  geom_boxplot() +
  geom_jitter(width = 0.2, height = 0)
```



Finally, we inspect the parameter estimates as a function of class.

```
class_params = read.table(text = "parameter x1 estimate_c1 x2 lower_c1 upper_c1 x3 estimate_c2 x21 lowe
        m_o = 0.600808  [  0.066186  1.135430 ]  1.000000  [  0.859549  1.140451 ]  1.000000  [  0.9373
          m_t = 0.581545  [  0.105257  1.057833 ]  0.000000  [ -0.146289  0.146289 ]  1.000000  [  0.93
            s = 1.000000  [ -4.257337  6.257337 ]  0.666724  [  0.385543  0.947906 ]  0.883616  [  0.72
 ", header = TRUE) %>%
  select(-starts_with("x")) %>%
  gather("what", "value", -parameter) %>%
  separate("what", c("measure", "class")) %>%
  spread(measure, value)

class_params %>%
  mutate_at(vars("estimate", "lower", "upper"),
            ~ifelse(. < -0.1, -0.1,
                    ifelse(. > 1.1, 1.1, .))) %>%
  ggplot(aes(x = fct_rev(factor(class)),
             y = estimate,
             ymin = lower,
             ymax = upper)) +
  geom_pointrange() +
  facet_wrap(~parameter) +
  scale_y_continuous(breaks = c(0, 0.5, 1)) +
  coord_flip() +
  labs(x = "class") +
```

```
  theme(panel.grid.major.x = element_line(),
        panel.grid.minor.x = element_line())
```



This shows that for class 1, all parameters are estimated with extreme imprecision. For classes 2 and 3, we see very high levels of m_o, a mirror pattern for m_t, and a monotonic increase of s, which is likely clearly above .5 for both classes.

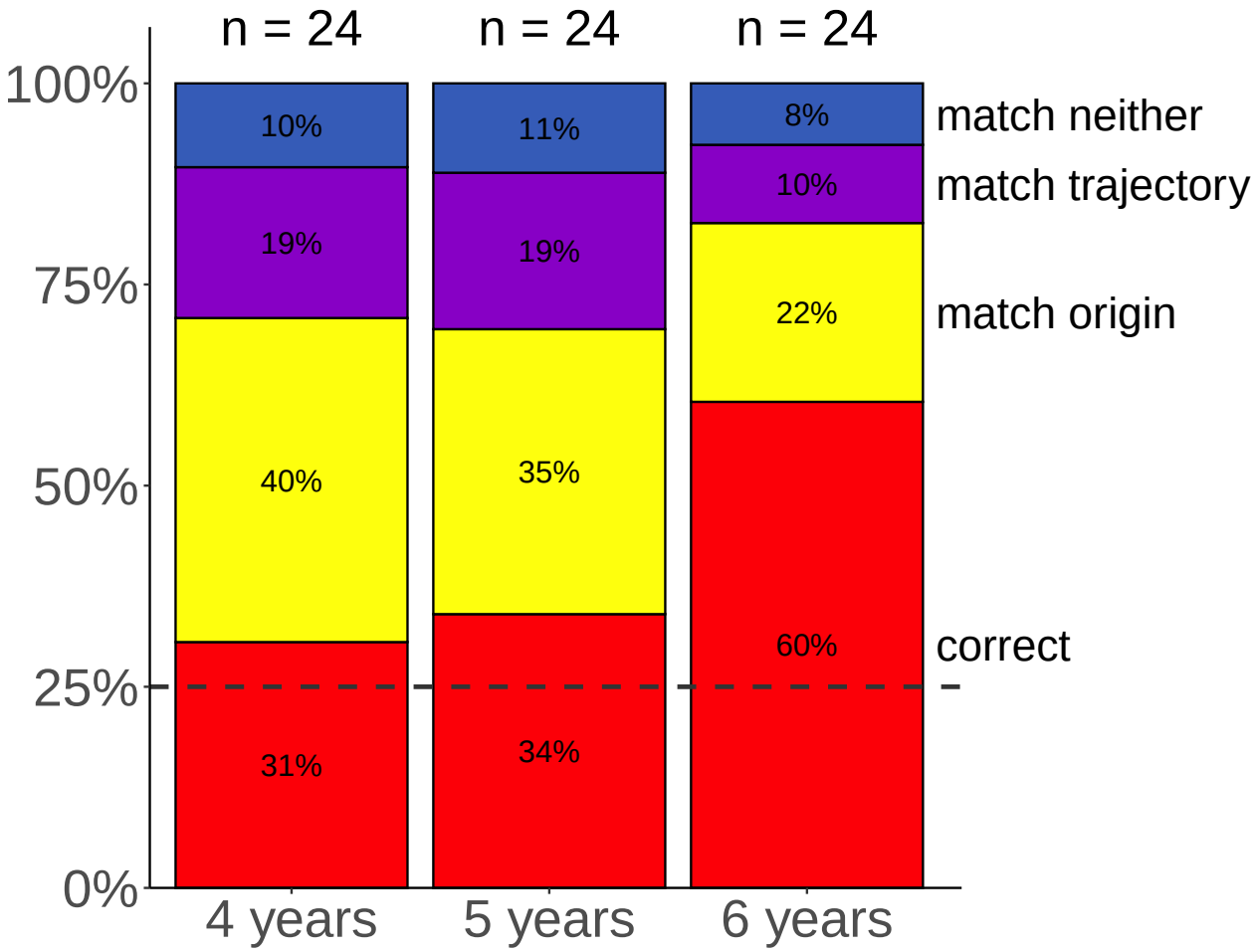
Plots

Stacked bar chart

```
df.plot = df.exp2 %>%
  mutate(answer = factor(answer,
                         labels = c("correct",
                                    "match origin",
                                    "match trajectory",
                                    "match neither")),
         question_index = str_c(outcome_actual, "\n",
                                outcome_counterfactual, "\n",
                                collision),
         question_index = as.factor(question_index)) %>%
  filter(collision == "collision") %>%
  count(age_group, answer) %>%
  group_by(age_group) %>%
  mutate(proportion = n / sum(n),
         label = str_c(round(proportion, 2) * 100, "%")) %>%
  ungroup()

df.text = df.exp2 %>%
  distinct(participant, age_group) %>%
```

```
  count(age_group) %>%
  mutate(x = 4:6,
         y = 1.07,
         label = str_c("n = ", n))

ggplot(data = df.plot,
       mapping = aes(x = age_group,
                     y = proportion,
                     fill = answer)) +
  geom_bar(stat = "identity",
           position = position_fill(reverse = TRUE),
           color = "black") +
  geom_hline(yintercept = 0.25,
             linetype = 2,
             size = 1,
             color = "gray20") +
  geom_text(mapping = aes(label = label),
            color = "black",
            # fontface = "bold",
            size = 5,
            position = position_stack(vjust = .5,
                                      reverse = TRUE)) +
  annotate(geom = "text",
           x = df.text$x,
           y = df.text$y,
           label = df.text$label,
           size = 8) +
  annotate(geom = "text",
           x = rep(6.5, 4),
           y = df.plot %>%
             filter(age_group == 6) %>%
             select(proportion) %>%
             mutate(cum_prop = cumsum(proportion),
                    lag = lag(cum_prop, 1, default = 0),
                    y = (cum_prop + lag) / 2) %>%
             pull(y),
           label = c("correct",
                     "match origin",
                     "match trajectory",
                     "match neither"),
           hjust = 0,
           size = 7) +
  labs(y = "% of responses") +
  scale_y_continuous(breaks = seq(0, 1, 0.25),
                     labels = str_c(seq(0, 100, 25), "%"),
                     expand = expansion(mult = 0)) +
  scale_x_continuous(breaks = 4:6,
                     labels = str_c(4:6, " years"),
                     expand = expansion(add = 0.1)) +
  coord_cartesian(clip = "off") +
  scale_fill_manual(values = c(rgb(252, 0, 8, maxColorValue = 255),
                               rgb(254, 255, 13, maxColorValue = 255),
                               rgb(135, 0, 197, maxColorValue = 255),
```

```
                              rgb(53, 91, 183, maxColorValue = 255))) +
  theme(text = element_text(size = 30),
        legend.position = "none",
        axis.title.y = element_blank(),
        axis.title.x = element_blank(),
        legend.title = element_blank(),
        plot.margin = margin(t = 0.5,
                             r = 5,
                             b = 0,
                             l = 0,
                             unit = "cm"))

ggsave("../figures//experiment2_bars.pdf",
       width = 8,
       height = 6)
```

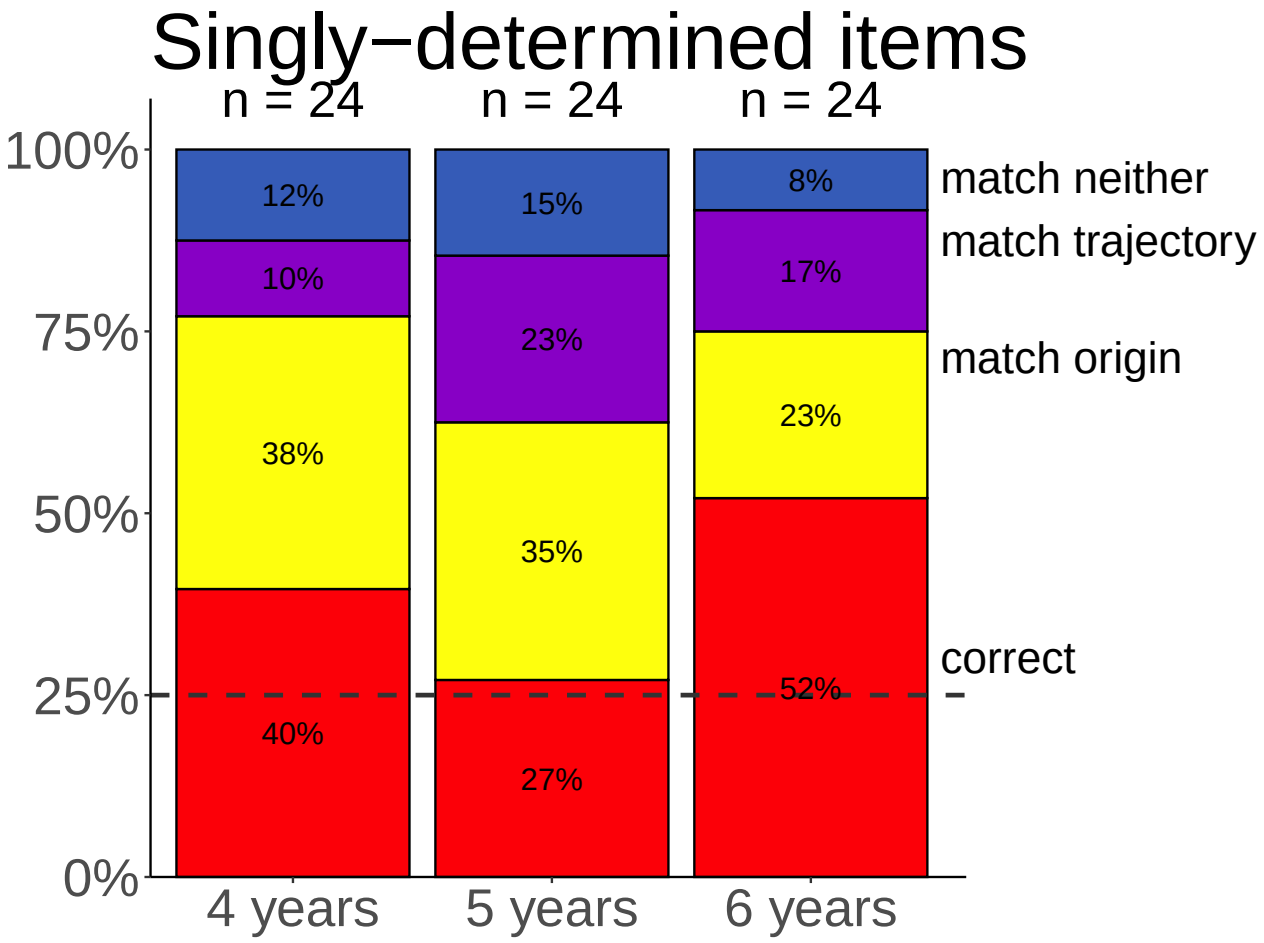


Let's also look at these results split up by singly-determined and over-determined

```
df.plot_singly_determined = df.exp2 %>%
  mutate(answer = factor(answer,
                         labels = c("correct",
                                    "match origin",
                                    "match trajectory",
                                    "match neither")),
```

```
        question_index = str_c(outcome_actual, "\n",
                               outcome_counterfactual, "\n",
                               collision),
        question_index = as.factor(question_index)) %>%
  filter(collision == "collision") %>%
  filter(outcome_counterfactual=="different") %>%
  count(age_group, answer) %>%
  group_by(age_group) %>%
  mutate(proportion = n / sum(n),
         label = str_c(round(proportion, 2) * 100, "%")) %>%
  ungroup()


ggplot(data = df.plot_singly_determined,
       mapping = aes(x = age_group,
                     y = proportion,
                     fill = answer)) +
  ggtitle("Singly-determined items") +
  geom_bar(stat = "identity",
           position = position_fill(reverse = TRUE),
           color = "black") +
  geom_hline(yintercept = 0.25,
             linetype = 2,
             size = 1,
             color = "gray20") +
  geom_text(mapping = aes(label = label),
            color = "black",
            # fontface = "bold",
            size = 5,
            position = position_stack(vjust = .5,
                                      reverse = TRUE)) +
  annotate(geom = "text",
           x = df.text$x,
           y = df.text$y,
           label = df.text$label,
           size = 8) +
  annotate(geom = "text",
           x = rep(6.5, 4),
           y = df.plot %>%
             filter(age_group == 6) %>%
             select(proportion) %>%
             mutate(cum_prop = cumsum(proportion),
                    lag = lag(cum_prop, 1, default = 0),
                    y = (cum_prop + lag) / 2) %>%
             pull(y),
           label = c("correct",
                     "match origin",
                     "match trajectory",
                     "match neither"),
           hjust = 0,
           size = 7) +
  labs(y = "% of responses") +
  scale_y_continuous(breaks = seq(0, 1, 0.25),
```

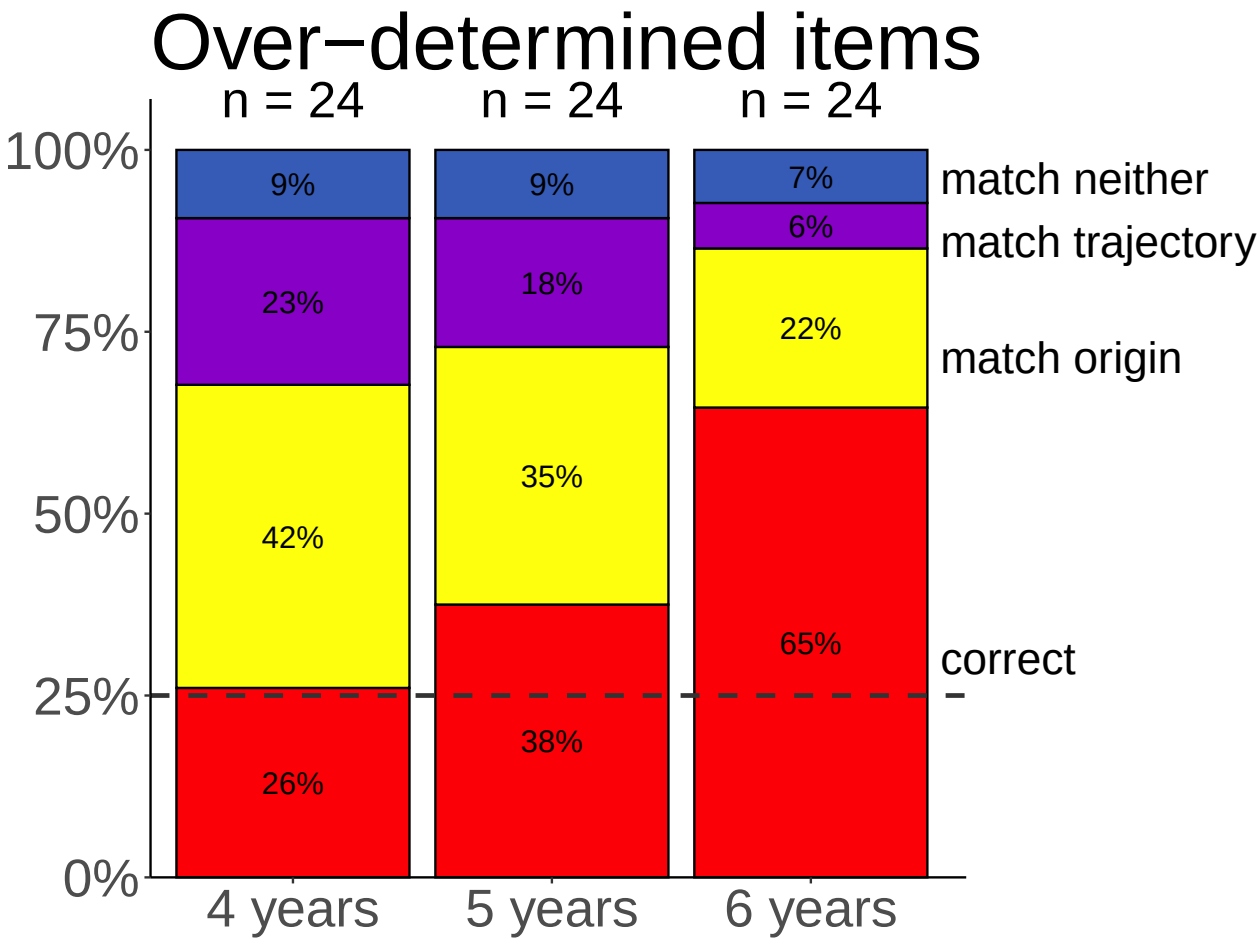
```
                     labels = str_c(seq(0, 100, 25), "%"),
                     expand = expansion(mult = 0)) +
  scale_x_continuous(breaks = 4:6,
                     labels = str_c(4:6, " years"),
                     expand = expansion(add = 0.1)) +
  coord_cartesian(clip = "off") +
  scale_fill_manual(values = c(rgb(252, 0, 8, maxColorValue = 255),
                               rgb(254, 255, 13, maxColorValue = 255),
                               rgb(135, 0, 197, maxColorValue = 255),
                               rgb(53, 91, 183, maxColorValue = 255))) +
  theme(text = element_text(size = 30),
        legend.position = "none",
        axis.title.y = element_blank(),
        axis.title.x = element_blank(),
        legend.title = element_blank(),
        plot.margin = margin(t = 0.5,
                             r = 5,
                             b = 0,
                             l = 0,
                             unit = "cm"))

ggsave("../figures//experiment2__singlydetermined_bars.pdf",
       width = 8,
       height = 6)
```




```
df.plot_over_determined = df.exp2 %>%
  mutate(answer = factor(answer,
                         labels = c("correct",
                                    "match origin",
                                    "match trajectory",
                                    "match neither")),
         question_index = str_c(outcome_actual, "\n",
                                outcome_counterfactual, "\n",
                                collision),
         question_index = as.factor(question_index)) %>%
  filter(collision == "collision") %>%
  filter(outcome_counterfactual=="same") %>%
  count(age_group, answer) %>%
  group_by(age_group) %>%
  mutate(proportion = n / sum(n),
         label = str_c(round(proportion, 2) * 100, "%")) %>%
  ungroup()


ggplot(data = df.plot_over_determined,
       mapping = aes(x = age_group,
                     y = proportion,
                     fill = answer)) +
  ggtitle("Over-determined items") +
```

```
geom_bar(stat = "identity",
         position = position_fill(reverse = TRUE),
         color = "black") +
geom_hline(yintercept = 0.25,
           linetype = 2,
           size = 1,
           color = "gray20") +
geom_text(mapping = aes(label = label),
          color = "black",
          # fontface = "bold",
          size = 5,
          position = position_stack(vjust = .5,
                                    reverse = TRUE)) +
annotate(geom = "text",
         x = df.text$x,
         y = df.text$y,
         label = df.text$label,
         size = 8) +
annotate(geom = "text",
         x = rep(6.5, 4),
         y = df.plot %>%
           filter(age_group == 6) %>%
           select(proportion) %>%
           mutate(cum_prop = cumsum(proportion),
                  lag = lag(cum_prop, 1, default = 0),
                  y = (cum_prop + lag) / 2) %>%
           pull(y),
         label = c("correct",
                   "match origin",
                   "match trajectory",
                   "match neither"),
         hjust = 0,
         size = 7) +
labs(y = "% of responses") +
scale_y_continuous(breaks = seq(0, 1, 0.25),
                   labels = str_c(seq(0, 100, 25), "%"),
                   expand = expansion(mult = 0)) +
scale_x_continuous(breaks = 4:6,
                   labels = str_c(4:6, " years"),
                   expand = expansion(add = 0.1)) +
coord_cartesian(clip = "off") +
scale_fill_manual(values = c(rgb(252, 0, 8, maxColorValue = 255),
                             rgb(254, 255, 13, maxColorValue = 255),
                             rgb(135, 0, 197, maxColorValue = 255),
                             rgb(53, 91, 183, maxColorValue = 255))) +
theme(text = element_text(size = 30),
      legend.position = "none",
      axis.title.y = element_blank(),
      axis.title.x = element_blank(),
      legend.title = element_blank(),
      plot.margin = margin(t = 0.5,
                           r = 5,
                           b = 0,
```

```
                              l = 0,
                              unit = "cm"))

ggsave("../figures//experiment2_overdetermined_bars.pdf",
       width = 8,
       height = 6)
```



Group-level parameteres

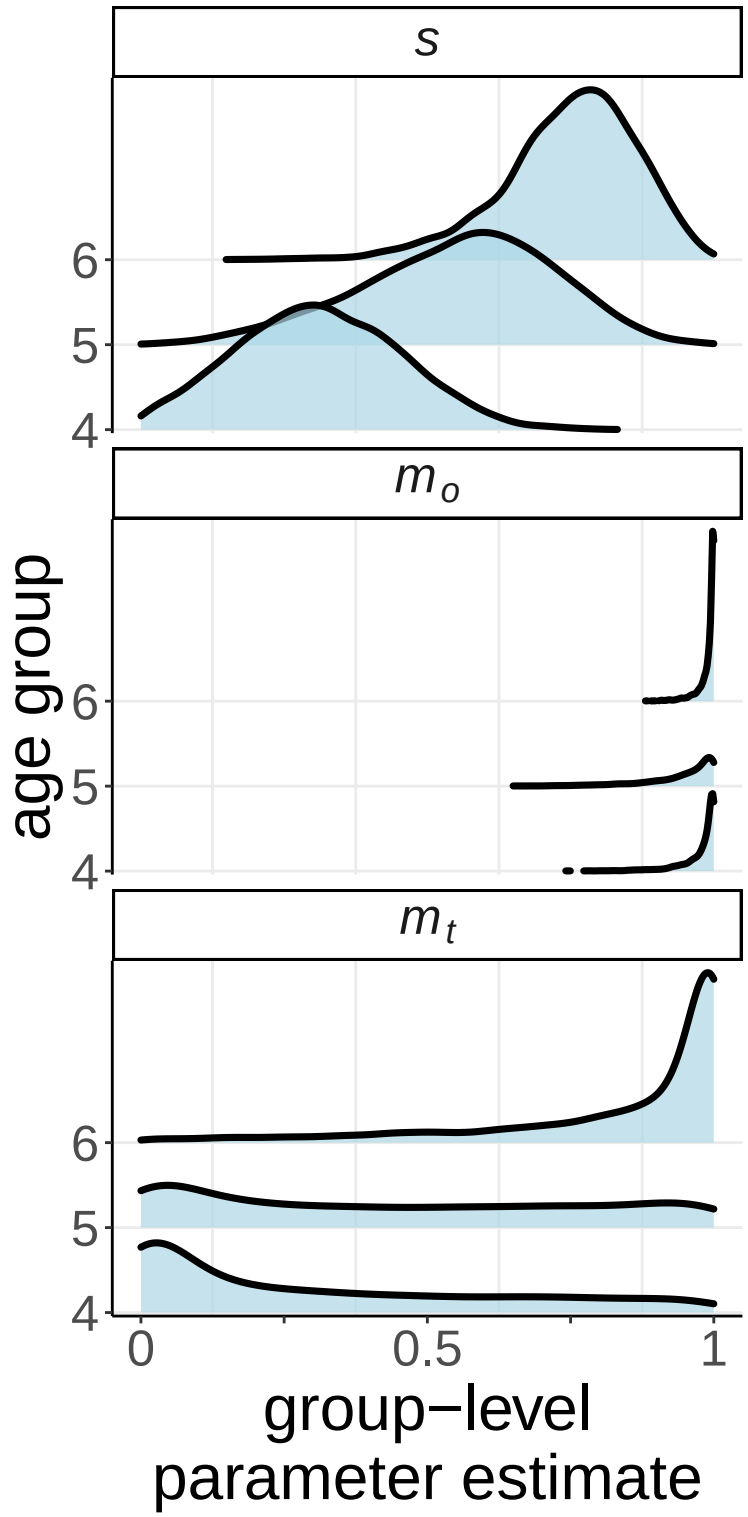
The following plot shows the posterior distribution of the group-level parameters, separated by age-group.

```
mus = gather_draws(fit_bayes_all$runjags, mu[i]) %>%
    mutate(
      parameter = factor(
        i,
        levels = c(3, 1, 2),
        labels = c("italic(s)", "italic(m[o])", "italic(m[t])"))
    ) %>%
    rename(mean = .value) %>%
    ungroup() %>%
    select(-i, -.variable)

pars = gather_draws(fit_bayes_all$runjags,
                    `factor_.*_age`[i], regex = TRUE) %>%
```

```
  ungroup() %>%
  mutate(age_group = factor(age_groups[i], level = age_groups),
         #estimate = pnorm(.value),
         parameter = factor(
           str_replace(str_replace(.variable, "factor_", ""), "_age", ""),
           levels = c("s", "m_o", "m_t"),
           labels = c("italic(s)", "italic(m[o])", "italic(m[t])"))) %>%
  mutate(age_group2 = factor(age_group, levels = rev(levels(di$age))))
pars = left_join(pars, mus) %>%
    mutate(estimate = pnorm(.value + mean))
```

```
#> Joining, by = c(".chain", ".iteration", ".draw", "parameter")
```

```
ggplot(pars, aes(x = estimate,
                 y = age_group,
                 # fill = ..x..,
                 height = ..density..)) +
  geom_density_ridges(scale = 2.0,
                      panel_scaling = TRUE,
                      rel_min_height = 0.001,
                      size = 1.25,
                      stat = "density",
                      bw = "nrd",
                      fill = "lightblue",
                      alpha = 0.7) +
  coord_cartesian(xlim = c(0, 1), ylim = c(1, 5.1)) +
  facet_wrap(~parameter,
             ncol = 1,
             labeller = label_parsed) +
  scale_y_discrete(expand = c(0.01, 0)) +
  scale_x_continuous(breaks = c(0, 0.25, 0.5, 0.75, 1),
                     labels = c("0", "", "0.5", "", "1")) +
  labs(y = "age group", x = "group-level\nparameter estimate") +
  # theme_ridges(center = TRUE) +
  theme(panel.grid.minor.x = element_line(),
        panel.grid.major.y = element_line())

ggsave("../figures//experiment2_estimates.pdf",
       width = 4,
       height = 8)
```


s
m_o
m_t
age group
6
5
4
0
0.5
1
group−level
parameter estimate

Experiment 3

Read in data

```
df.exp3 = read_csv("../data/experiment3_data.csv") %>%
  clean_names() %>%
  mutate(train_acc = ifelse(train_q == 2, 1, 0),
         test_acc = ifelse(test == 2, 1, 0)) %>%
  select(participant = snum, loc:test_acc)
```

Stats

Binomial test

```
df.results = df.exp3 %>%
  summarize(n = n(),
            n_correct = sum(test_acc == 1))

binom.test(x = df.results$n_correct,
           n = df.results$n) %>%
  tidy() %>%
  print_table(digits = 3)
```

estimate	statistic	p.value	parameter	conf.low	conf.high	method	alternative
0.667	20	0.099	30	0.472	0.827	Exact binomial test	two.sided

MPT model analysis

```
data = read.csv("../data/experiment3_data_individual.csv")
# 1 = incorrect 2 = correct.
data.coded = data %>%
  mutate(TrainAcc = ifelse(TrainQ == 2, 1, 0),
         TrainErr = ifelse(TrainQ == 1, 1, 0),
         TestAcc = ifelse(Test == 2, 1, 0),
         TestErr = ifelse(Test == 1, 1, 0))

data.agg = data.coded %>%
  summarize(corr=sum(TestAcc),err=sum(TestErr))
```

MPT analysis

First, we get a simple point estimate.

```
fit1 = fit.mpt(data.agg, "models/mpt_exp3.txt")
```

```
#> [1] "Model fitting begins at 2020-10-11 13:30:40"
#> [1] "Model fitting stopped at 2020-10-11 13:30:40"
#> Time difference of 0.002117157 secs
```

```
fit1$goodness.of.fit
```

```
#>   Log.Likelihood     G.Squared df p.value
#> 1      -19.09543 -1.421085e-14  0       1
```

```
fit1$parameters
```

```
#>   estimates  lower.conf upper.conf
#> s 0.3333333 -0.00404035   0.670707
```

```
# raw parameter estimate of $s$ is .333
```

However, to be comparable with the previous analyses, we need to conduct a Bayesian analysis of the aggregated data. Because we only have one trial per child, we opted for the Bayesian model on the aggregated data in this case. This assumes that individual variability is consistent with the variability of a multinomial distribution.

```
data.agg2 = data.coded %>%
  summarize(TestAcc = sum(TestAcc),
            TestErr = sum(TestErr))

smpt = simpleMPT("models/model3.eqn",
                 data.agg2,
                 restrictions = list("g=0.5"),
                 n.chains = 4)
summary(smpt)
```

```
#> Call:
#> simpleMPT(eqnfile = "models/model3.eqn", data = data.agg2, restrictions = list("g=0.5"),
#>     n.chains = 4)
#>
#>         Mean   SD  2.5%   50% 97.5% Time-series SE n.eff  Rhat R_95%
#> mean_s 0.328 0.15 0.055 0.327 0.606             NA    NA 1.001 1.005
#>
#>
#> ###############
#>  Model fit statistics (posterior predictive p-values):
#> Use PPP(fittedModel) to get T1 and T2 posterior predictive checks.
```

```
str(smpt, 2)
```

```
#> List of 7
#>  $ summary  :List of 6
#>   ..$ groupParameters      :List of 6
#>   ..$ individParameters    : num [1, 1, 1:9] 0.3276 0.15 0.0554 0.3275 0.6063 ...
#>   .. ..- attr(*, "dimnames")=List of 3
#>   ..$ fitStatistics        : NULL
#>   ..$ transformedParameters: NULL
#>   ..$ call                 : chr "(summarizeMPT called manually)"
#>   ..$ round                : num 3
#>   ..- attr(*, "class")= chr "summary.simpleMPT"
#>  $ mptInfo  :List of 15
#>   ..$ model                : chr "simpleMPT"
#>   ..$ thetaNames           :'data.frame':    1 obs. of  2 variables:
#>   ..$ thetaUnique          : chr "s"
#>   ..$ thetaFixed           : NULL
#>   ..$ MPT                  :List of 13
#>   ..$ eqnfile              : chr "models/model3.eqn"
#>   ..$ data                 :'data.frame':    1 obs. of  2 variables:
#>   ..$ restrictions         :List of 1
#>   ..$ covData              : NULL
```

```
#>   ..$ corProbit            : logi TRUE
#>   ..$ predTable            : NULL
#>   ..$ predFactorLevels     : NULL
#>   ..$ predType             : NULL
#>   ..$ transformedParameters: NULL
#>   ..$ hyperprior           :List of 2
#>  $ mcmc.summ: num [1:3, 1:9] 0.328 NaN 0.328 0.15 NA ...
#>   ..- attr(*, "dimnames")=List of 2
#>  $ runjags  :List of 1
#>   ..$ mcmc:List of 4
#>   .. ..- attr(*, "class")= chr "mcmc.list"
#>  $ postpred : NULL
#>  $ call     : language simpleMPT(eqnfile = "models/model3.eqn", data = data.agg2, restrictions = lis
#>  $ time     : 'difftime' num 0.0836548805236816
#>   ..- attr(*, "units")= chr "secs"
#>  - attr(*, "class")= chr "simpleMPT"
```

```
head(smpt$runjags$mcmc)
```

```
#> [[1]]
#> Markov Chain Monte Carlo (MCMC) output:
#> Start = 501
#> End = 519
#> Thinning interval = 3
#>           mean sd theta[1,1]
#> [1,] 0.2050122 NA  0.2050122
#> [2,] 0.3108400 NA  0.3108400
#> [3,] 0.4835955 NA  0.4835955
#> [4,] 0.1696628 NA  0.1696628
#> [5,] 0.3475230 NA  0.3475230
#> [6,] 0.1680781 NA  0.1680781
#> [7,] 0.1908167 NA  0.1908167
#>
#> [[2]]
#> Markov Chain Monte Carlo (MCMC) output:
#> Start = 501
#> End = 519
#> Thinning interval = 3
#>           mean sd theta[1,1]
#> [1,] 0.2340487 NA  0.2340487
#> [2,] 0.2258720 NA  0.2258720
#> [3,] 0.6467904 NA  0.6467904
#> [4,] 0.5312226 NA  0.5312226
#> [5,] 0.5538257 NA  0.5538257
#> [6,] 0.2844771 NA  0.2844771
#> [7,] 0.3100316 NA  0.3100316
#>
#> [[3]]
#> Markov Chain Monte Carlo (MCMC) output:
#> Start = 501
#> End = 519
#> Thinning interval = 3
#>           mean sd theta[1,1]
#> [1,] 0.2364754 NA  0.2364754
#> [2,] 0.4020819 NA  0.4020819
```

```
#> [3,] 0.2835038 NA  0.2835038
#> [4,] 0.5199267 NA  0.5199267
#> [5,] 0.4319134 NA  0.4319134
#> [6,] 0.3193283 NA  0.3193283
#> [7,] 0.3279902 NA  0.3279902
#>
#> [[4]]
#> Markov Chain Monte Carlo (MCMC) output:
#> Start = 501
#> End = 519
#> Thinning interval = 3
#>           mean sd theta[1,1]
#> [1,] 0.2326497 NA  0.2326497
#> [2,] 0.3131550 NA  0.3131550
#> [3,] 0.2867504 NA  0.2867504
#> [4,] 0.2191924 NA  0.2191924
#> [5,] 0.1463435 NA  0.1463435
#> [6,] 0.1360975 NA  0.1360975
#> [7,] 0.6335324 NA  0.6335324
#>
#> attr(,"class")
#> [1] "mcmc.list"
```

```
samps = gather_draws(smpt$runjags$mcmc, mean)
```

Here are the mode and mean of the estimate of s, and their 80% HDIs, based on this analysis.

Mode (reported, for comparison w/Exp 2 4-year-olds):

```
samps %>%
  mode_hdci(.value,
            .width = c(0.80))
```

```
#> # A tibble: 1 x 7
#>   .variable .value .lower .upper .width .point .interval
#>   <chr>      <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
#> 1 mean       0.333 0.0960  0.500    0.8 mode   hdci
```

Mean:

```
samps %>%
  mean_hdci(.value,
            .width = c(0.80))
```

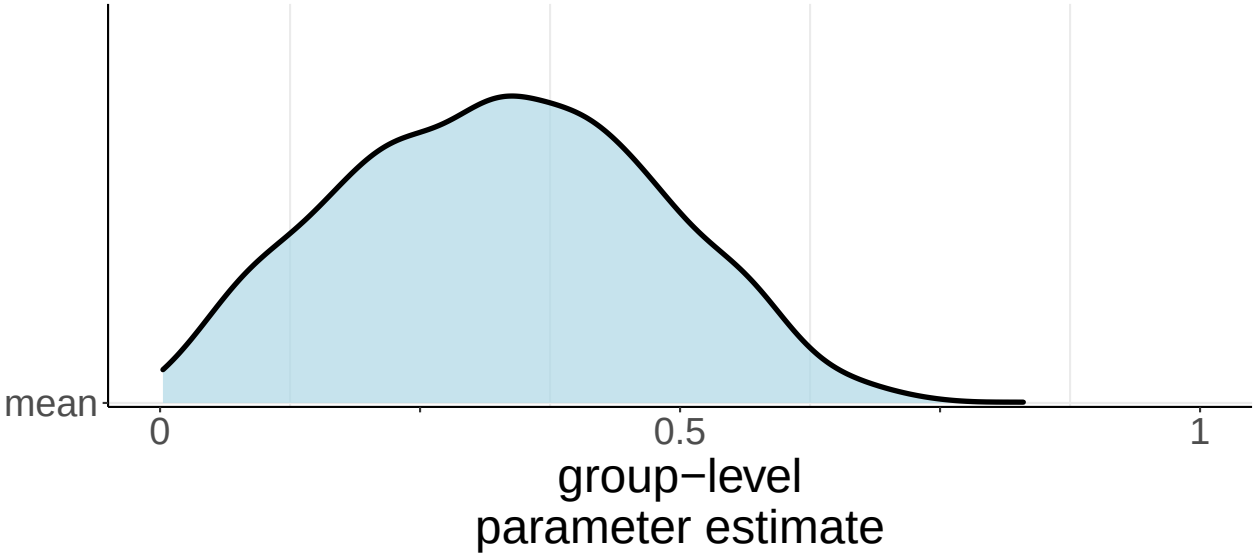
```
#> # A tibble: 1 x 7
#>   .variable .value .lower .upper .width .point .interval
#>   <chr>      <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
#> 1 mean       0.328 0.0960  0.500    0.8 mean   hdci
```

Plots

Density plot of the the estimate.

```
ggplot(samps,
       aes(x = .value,
           y = .variable,
           height = ..density..)) +
  geom_density_ridges(scale = 2.0,
```

```
                      panel_scaling = TRUE,
                      rel_min_height = 0.001,
                      size = 1.25,
                      stat = "density",
                      bw = "nrd",
                      fill = "lightblue",
                      alpha = 0.7) +
  coord_cartesian(xlim = c(0, 1), ylim = c(1, 7.1)) +
  scale_y_discrete(expand = c(0.01, 0)) +
  scale_x_continuous(breaks = c(0, 0.25, 0.5, 0.75, 1),
                     labels = c("0", "", "0.5", "", "1")) +
  labs(y = element_blank(), x = "group-level\nparameter estimate") +
  theme(panel.grid.minor.x = element_line(),
        panel.grid.major.y = element_line())

ggsave("../figures/experiment3_estimate.pdf",
       width = 4,
       height = 4)
```



Session info

```
sessionInfo()
```

```
#> R version 3.6.3 (2020-02-29)
#> Platform: x86_64-apple-darwin15.6.0 (64-bit)
#> Running under: macOS Catalina 10.15.7
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
#> LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
#>
#> locale:
#> [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
#>
#> attached base packages:
```

```
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] forcats_0.5.0          stringr_1.4.0          dplyr_1.0.2
#>  [4] purrr_0.3.4            readr_1.4.0            tidyr_1.1.2
#>  [7] tibble_3.0.3           ggplot2_3.3.2          tidyverse_1.3.0
#> [10] patchwork_1.0.1        afex_0.28-0            lme4_1.1-23
#> [13] MPTinR_1.13.0          ggridges_0.5.2         tidybayes_2.1.1
#> [16] TreeBUGS_1.4.5         BayesFactor_0.9.12-4.2 Matrix_1.2-18
#> [19] coda_0.19-4            ggeffects_0.16.0       broom_0.7.1
#> [22] xtable_1.8-4           kableExtra_1.2.1       knitr_1.30
#> [25] janitor_2.0.1
#>
#> loaded via a namespace (and not attached):
#>   [1] readxl_1.3.1         backports_1.1.10     Hmisc_4.4-1
#>   [4] plyr_1.8.6           splines_3.6.3        svUnit_1.0.3
#>   [7] elliptic_1.4-0       TH.data_1.0-10       digest_0.6.25
#>  [10] htmltools_0.5.0      lmerTest_3.1-2       fansi_0.4.1
#>  [13] checkmate_2.0.0      magrittr_1.5         contfrac_1.1-12
#>  [16] cluster_2.1.0        openxlsx_4.2.2       modelr_0.1.8
#>  [19] sandwich_3.0-0       jpeg_0.1-8.1         colorspace_1.4-1
#>  [22] blob_1.2.1           rvest_0.3.6          ggdist_2.2.0
#>  [25] haven_2.3.1          xfun_0.18            crayon_1.3.4
#>  [28] jsonlite_1.7.1       survival_3.2-7       zoo_1.8-8
#>  [31] glue_1.4.2           gtable_0.3.0         emmeans_1.5.1
#>  [34] webshot_0.5.2        MatrixModels_0.4-1   distributional_0.2.1
#>  [37] car_3.0-10           abind_1.4-5          scales_1.1.1
#>  [40] mvtnorm_1.1-1        GGally_2.0.0         DBI_1.1.0
#>  [43] Rcpp_1.0.5           htmlTable_2.1.0      viridisLite_0.3.0
#>  [46] tmvnsim_1.0-2        HDInterval_0.2.2     foreign_0.8-75
#>  [49] deSolve_1.28         Formula_1.2-3        htmlwidgets_1.5.2
#>  [52] httr_1.4.2           arrayhelpers_1.1-0   RColorBrewer_1.1-2
#>  [55] runjags_2.0.4-6      logspline_2.1.16     ellipsis_0.3.1
#>  [58] reshape_0.8.8        pkgconfig_2.0.3      farver_2.0.3
#>  [61] nnet_7.3-14          dbplyr_1.4.4         utf8_1.1.4
#>  [64] labeling_0.3         tidyselect_1.1.0     rlang_0.4.8
#>  [67] reshape2_1.4.4       munsell_0.5.0        cellranger_1.1.0
#>  [70] tools_3.6.3          cli_2.0.2            generics_0.0.2
#>  [73] sjlabelled_1.1.7     evaluate_0.14        yaml_2.2.1
#>  [76] fs_1.5.0             zip_2.1.1            hypergeo_1.2-13
#>  [79] pbapply_1.4-3        nlme_3.1-149         xml2_1.3.2
#>  [82] compiler_3.6.3       rstudioapi_0.11      curl_4.3
#>  [85] png_0.1-7            reprex_0.3.0         statmod_1.4.34
#>  [88] stringi_1.5.3        Brobdingnag_1.2-6    lattice_0.20-41
#>  [91] psych_2.0.9          nloptr_1.2.2.2       vctrs_0.3.4
#>  [94] pillar_1.4.6         lifecycle_0.2.0      estimability_1.3
#>  [97] data.table_1.12.8    insight_0.9.6        R6_2.4.1
#> [100] latticeExtra_0.6-29  gridExtra_2.3        rio_0.5.16
#> [103] rjags_4-10           codetools_0.2-16     boot_1.3-25
#> [106] MASS_7.3-53          gtools_3.8.2         assertthat_0.2.1
#> [109] withr_2.3.0          mnormt_2.0.2         multcomp_1.4-14
#> [112] parallel_3.6.3       hms_0.5.3            grid_3.6.3
#> [115] rpart_4.1-15         minqa_1.2.4          rmarkdown_2.4
#> [118] snakecase_0.11.0     carData_3.0-4        numDeriv_2016.8-1.1
```

```
#> [121] lubridate_1.7.9      base64enc_0.1-3
```