practical_exercise_3, Methods 3, 2021, autumn semester

Sirid Wihlborg

04/10/21

Exercise 1

Files downloaded from 'experiment 2': https://osf.io/ecxsj/files/ The data is associated with Experiment 2 of the article at the following DOI https://doi.org/10.1016/j.concog.2019.03.007

1) Put the data from all subjects into a single data frame

```
list <- list.files(path = "data/experiment_2", pattern = "*.csv", full.names=TRUE) # importing all files
df <- ldply(list, read_csv) # making them into one data-frame
```

2) Describe the data and construct extra variables from the existing variables

The dataset contains 18131 observations described by 17 variables. Data from 29 subjects is included.

i. add a variable to the data frame and call it _correct_ (have it be a _logical_ variable). Assign a 1

```
df <- df %>%
  mutate(correct = ifelse(target.type == "even" & obj.resp == "e" |
                            target.type == "odd" & obj.resp == "o", 1, 0))
```

ii. describe what the following variables in the data frame contain, _trial.type_, _pas_, _trial_, _targ

```
df <- df %>%
  select(trial.type, pas, trial, target.contrast, cue, task, target.type, rt.subj, rt.obj, obj.resp, sub
```

"trial.type" (character): indicate if the participant did the first experiment ('staircase') or the follow-up study ('experiment'). "pas" (numeric): perceptual awareness scale where the four levels of rating are numbered 1:4 (hence class numeric) on a scale ranging from "no experience" to "full experience". "trial (numeric): number indicating trial number. A numbered list for every trial the subject completes, i.e. presses e or o in either of the trial types., per subject."target.contrast" (numeric): The contrast between the background and the digit (stimuli). A value between 0-1 hence numeric. "cue" (factor?): The specific cue pattern, I'm not really sure how and why to classify this. "task" (character?): Whether cue pattern is 2 (singles), 4 (pairs) or 8 (quadruplets) digits. I'm not really sure how and why to classify this. "target-type" (character): text indicating if the stimuli-number was even or odd ("even"/"odd"). "rt.subj" (numeric): Reaction time for response to PAS pr. trail "rt.obj" (numeric): Reaction time for responding if target is even or odd "obj.resp" (character): letters indicating what response the participant gave to the stimulus ("o" = odd / "e" = even) "subject" (factor?): a number specific to each participant. I'm not really sure how and why to classify this. "correct" (logical): a number indicating if the participant was right in judging the stimuli (1 = correct, 0 = false)

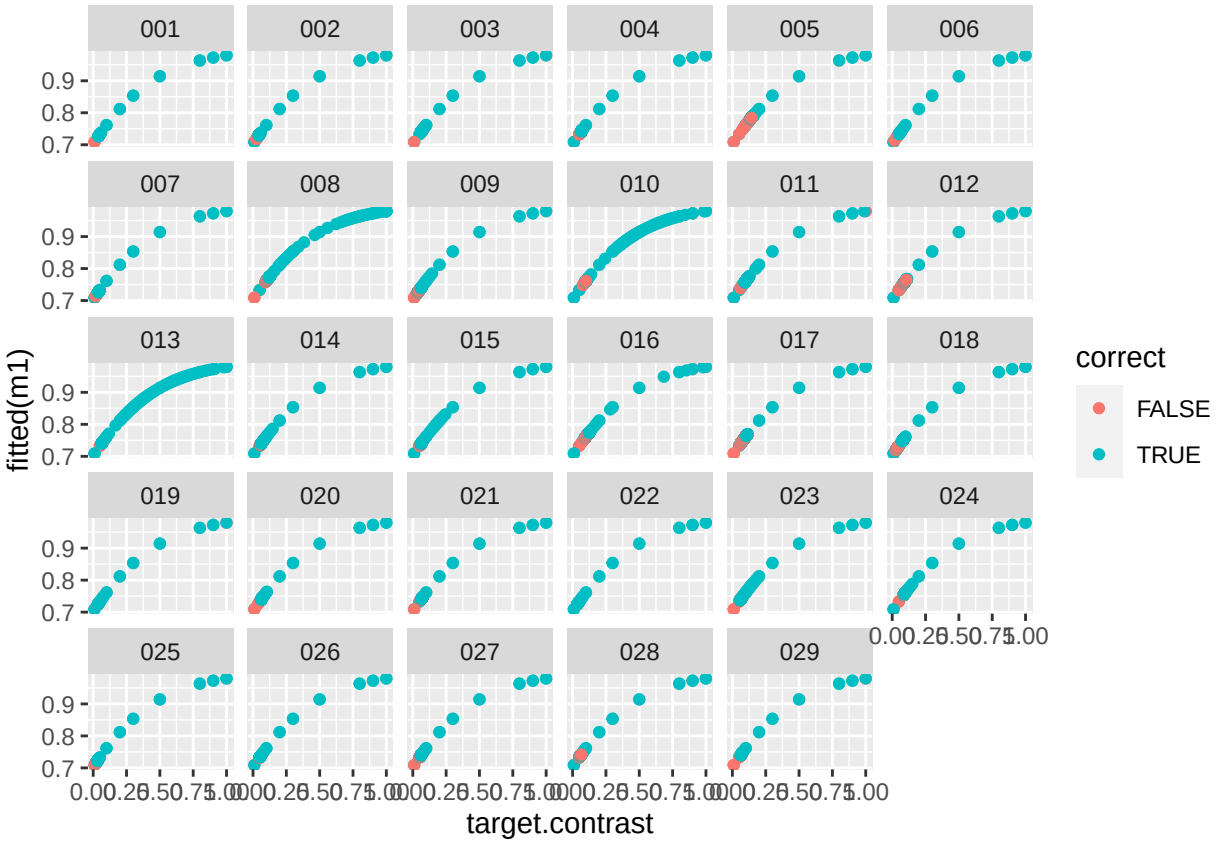
```
# making the correct column logical
df$correct <- as.logical(df$correct)
```

iii. for the staircasing part __only__, create a plot for each subject where you plot the estimated fun

```
df_staircase <- df %>%
  filter(trial.type == "staircase")

m1 <- glm(correct ~ target.contrast, data = df_staircase, family = "binomial")

ggplot(data = df_staircase, aes(x = target.contrast, y = fitted(m1), color = correct)) +
  geom_point() +
  facet_wrap( ~ subject)
```

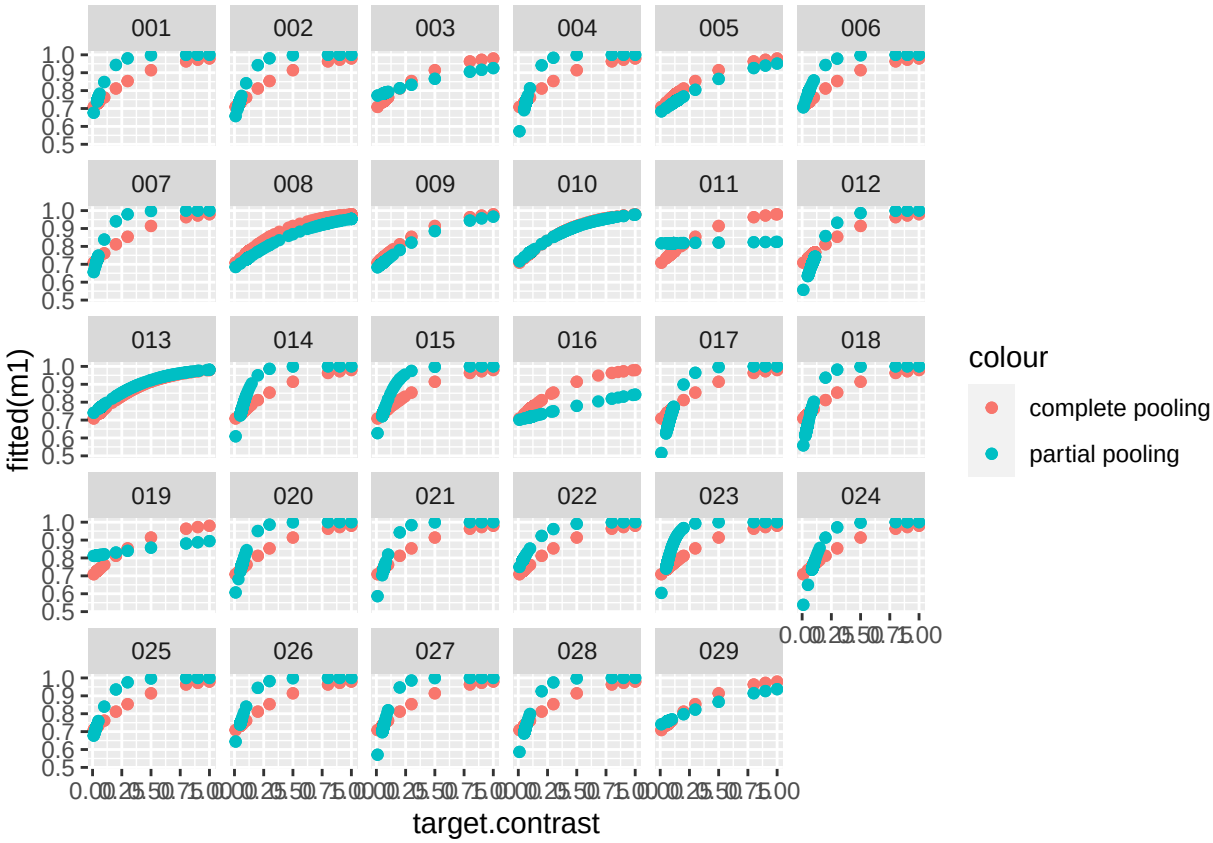


Having *correct* only being dependent on *target-contrast* I would assume will make a *complete pooling* model, since we take no account of individual differences.

iv. on top of those plots, add the estimated functions (on the _target.contrast_ range from 0-1) for ea

```
m2 <- lme4::glmer(correct ~ target.contrast + (1+target.contrast|subject), data = df_staircase, family =

ggplot(data = df_staircase) +
  geom_point(aes(x = target.contrast, y = fitted(m1), color = "complete pooling")) +
  geom_point(aes(x = target.contrast, y = fitted(m2), color = "partial pooling")) +
  facet_wrap( ~ subject)
```



`v. in your own words, describe how the partial pooling model allows for a better fit for each subject`

Since people are different and have different cognitive abilities, they will often perform very different in tasks including tasks in experimental settings. If you take the average performance and/or effect to represent every subject, you are bound to the failure that some participants will be way above and/or below average, which is clearly illustrated in the plot above. By making a mixed effect model (partial pooling) we tell the model that different people have different performance and because the model "knows" this, it's much better at giving the correct fit to each participant.

Exercise 2

Now we **only** look at the *experiment* trials (*trial.type*)

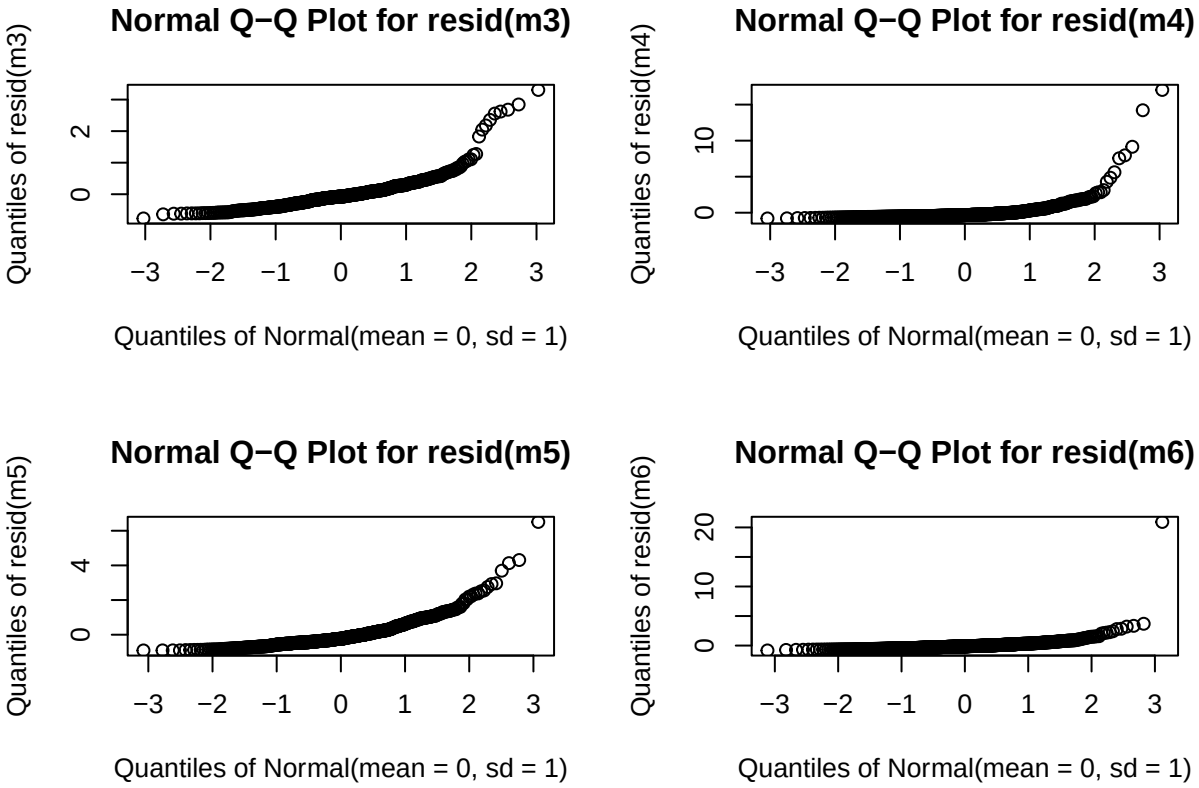
```
df_experiment <- df %>%
  filter(trial.type == "experiment")
```

1) Pick four subjects and plot their Quantile-Quantile (Q-Q) plots for the residuals of their objective response times (*rt.obj*) based on a model where only intercept is modelled

```
df_subject1 <- df_experiment[which(df$subject == "001"),]
df_subject2 <- df_experiment[which(df$subject == "002"),]
df_subject3 <- df_experiment[which(df$subject == "003"),]
df_subject4 <- df_experiment[which(df$subject == "004"),]
```

```r
m3 <- lm(rt.obj ~ 1, data = df_subject1)
m4 <- lm(rt.obj ~ 1, data = df_subject2)
m5 <- lm(rt.obj ~ 1, data = df_subject3)
m6 <- lm(rt.obj ~ 1, data = df_subject4)

par(mfrow=c(2,2))
qqPlot(resid(m3))
qqPlot(resid(m4))
qqPlot(resid(m5))
qqPlot(resid(m6))
```



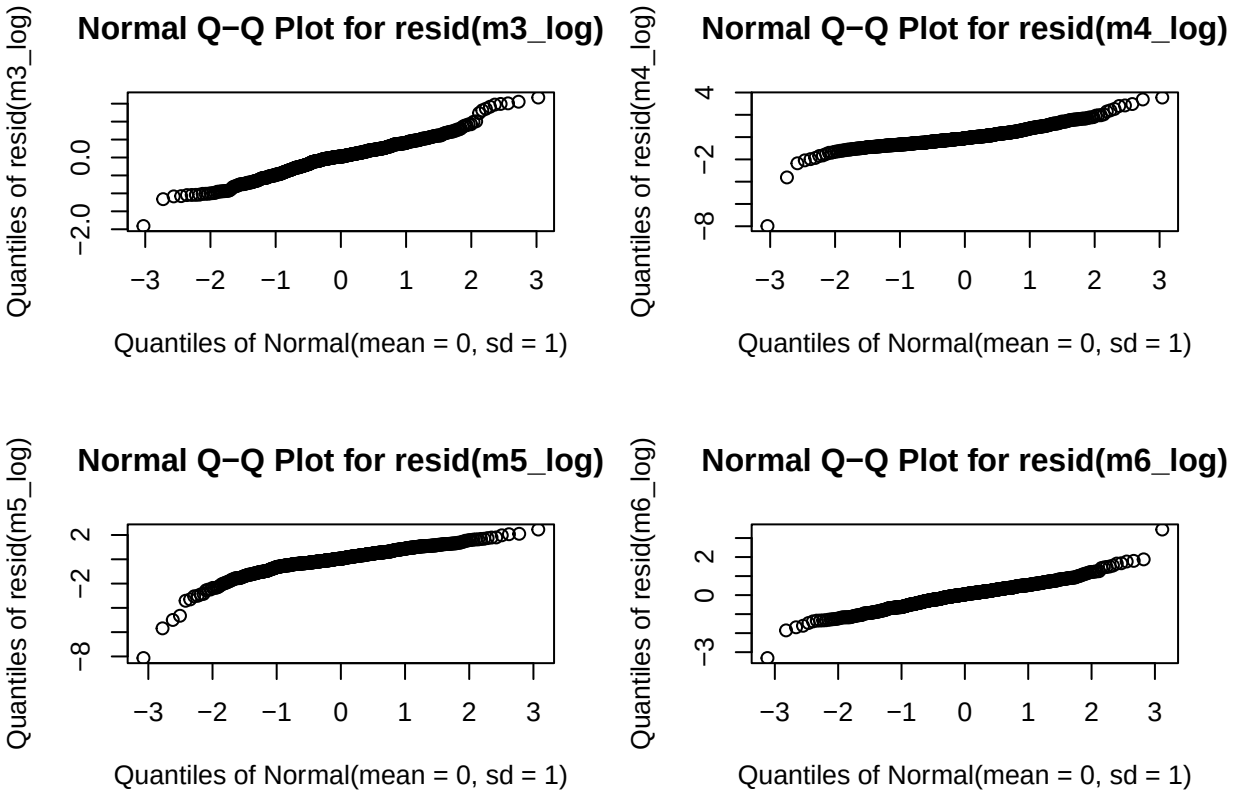
i. comment on these

The qqplots for the first three participants indicate that the residuals are quite rightskewed and not normally distributed. For participant 4 it actually looks a bit more normally distributed, however a far out right outlier makes it not-normal as well.

ii. does a log-transformation of the response time data improve the Q-Q-plots?

```r
m3_log <- lm(log(rt.obj) ~ 1, data = df_subject1)
m4_log <- lm(log(rt.obj) ~ 1, data = df_subject2)
m5_log <- lm(log(rt.obj) ~ 1, data = df_subject3)
m6_log <- lm(log(rt.obj) ~ 1, data = df_subject4)
```

```
par(mfrow=c(2,2))
qqPlot(resid(m3_log))
qqPlot(resid(m4_log))
qqPlot(resid(m5_log))
qqPlot(resid(m6_log))
```



For participant 1, 2 and 4 the residuals are definitely more normally distributed. QQplot from Participant 3 however reveals a left-skewed distribution of residuals.

2) Now do a partial pooling model modelling objective response times as dependent on *task*? (set `REML=FALSE` in your `lmer`-specification)

   i. which would you include among your random effects and why? (support your choices with relevant measures, taking into account variance explained and number of parameters going into the modelling)

I would sure have subject as random intercept as I except different baseline levels pr. participant. Allowing the performance in a given task to vary (ie. having *task* as random slope) gives a boundary-(singular)-error, hence making the model too complex, so I'll leave that out. I tried to allow different baseline pr. task (ie. having *task* as random intercept) but still got the error telling me my model is too complex. When modelling this I also saw that *task* explained 0 variance, therefore it's not quite useful.

I generally think there's too little data to make a robust models with many parameters, therefore my final model is quite simple only having subject as random intercept:

```
m7 <- lmerTest::lmer(rt.obj ~ task + (1|subject), data = df_experiment, REML = FALSE)
```

ii. explain in your own words what your chosen models says about response times between the different ta

Below you see the coefficients for the fixed effects. By using 'lmer' from the 'lmerTest'-package I got p-values, and I conclude that p-values were $< .05$. I can conclude that the average response time for task = pairs ('(Intercept)' value) was 1.12 sec. We then see that rt for task = quadruplet is 0.15 sec. slower and rt for task = singles is furthermore 0.19 sec. slower.

We hence see that there's difference in response time across condition, though the difference is not super large.

```
fixef(m7)
```

```
##    (Intercept) taskquadruplet    tasksingles
##      1.1200841     -0.1532473     -0.1915364
```

3) Now add *pas* and its interaction with *task* to the fixed effects

```
m8 <- lmerTest::lmer(rt.obj ~ task * pas + (1|subject), data = df_experiment, REML = FALSE)
```

i. how many types of group intercepts (random effects) can you add without ending up with convergence is

```
m9 <- lmerTest::lmer(rt.obj ~ task * pas + (1|subject) + (1|task) + (1|pas), data = df_experiment, REML
```

```
## boundary (singular) fit: see ?isSingular
```

```
m10 <- lmerTest::lmer(rt.obj ~ task * pas + (1|subject) + (1|task), data = df_experiment, REML = FALSE)
```

```
## boundary (singular) fit: see ?isSingular
```

```
m11 <- lmerTest::lmer(rt.obj ~ task * pas + (1|subject) + (1|pas), data = df_experiment, REML = FALSE)
```

I can add two random intercepts: one for subject and one for pas, without getting the singular fit error :)

ii. create a model by adding random intercepts (without modelling slopes) that results in a singular fi

Well, since my above-made model 'm10' gave me the error, I'll use that.

```
print(VarCorr(m10), comp='Variance')
```

```
##  Groups   Name        Variance
##  subject  (Intercept) 0.097438
##  task     (Intercept) 0.000000
##  Residual             8.153792
```

iii. in your own words - how could you explain why your model would result in a singular fit?

I see that adding 'task' as random intercept explains '0' variance, which is why I get the error. I would assume that this is because 'task' is highly correlated with one or more of the other random effects, in this case 'subject'. since subject is the only other random effect in my model.

Exercise 3

1) Initialise a new data frame, `data.count`. *count* should indicate the number of times they categorized their experience as *pas* 1-4 for each *task*. I.e. the data frame would have for subject 1: for task:singles, pas1 was used # times, pas2 was used # times, pas3 was used # times and pas4 was used # times. You would then do the same for task:pairs and task:quadruplet

```
data.count <- df %>%
  group_by(subject, task, pas) %>%
  dplyr::summarise("count" = n())
```

```
## `summarise()` has grouped output by 'subject', 'task'. You can override using the `.groups` argument
```

2) Now fit a multilevel model that models a unique "slope" for *pas* for each *subject* with the interaction between *pas* and *task* and their main effects being modelled

```
m12 <- lme4::glmer(count ~ pas * task + (pas|subject), data = data.count, family = poisson)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00689826 (tol = 0.002, component 1)
```

```
summary(m12)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: count ~ pas * task + (pas | subject)
##    Data: data.count
##
##      AIC      BIC   logLik deviance df.resid
##   4685.1   4719.6  -2333.6   4667.1      331
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.7718 -1.9208 -0.1275  1.6133 11.6478
##
## Random effects:
##  Groups  Name        Variance Std.Dev. Corr
##  subject (Intercept) 1.2016   1.0962
##          pas         0.2203   0.4694   -0.99
## Number of obs: 340, groups:  subject, 29
##
## Fixed effects:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)         4.29023    0.20576  20.851  < 2e-16 ***
## pas                -0.19516    0.08798  -2.218   0.0265 *
## taskquadruplet      0.16667    0.04007   4.160 3.19e-05 ***
## tasksingles        -0.39664    0.04192  -9.462  < 2e-16 ***
## pas:taskquadruplet -0.07193    0.01606  -4.479 7.50e-06 ***
## pas:tasksingles     0.16857    0.01587  10.623  < 2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) pas    tskqdr tsksng ps:tskq
## pas         -0.989
## taskqudrplt -0.100  0.083
## tasksingles -0.096  0.078  0.490
## ps:tskqdrpl  0.088 -0.091 -0.891 -0.430
## ps:tsksngls  0.089 -0.091 -0.456 -0.900  0.501
## optimizer (Nelder_Mead) convergence code: 0 (OK)
## Model failed to converge with max|grad| = 0.00689826 (tol = 0.002, component 1)
```

i. which family should be used?

Poisson. Since we're dealing with 'counts' over a series of time.

ii. why is a slope for _pas_ not really being modelled?

iii. if you get a convergence error, try another algorithm (the default is the _Nelder_Mead_) - try (_bo

```
m13 <- glmer(count ~ pas * task + (pas|subject), data = data.count, family = poisson, control = glmerCon
```

iv. when you have a converging fit - fit a model with only the main effects of _pas_ and _task_. Compare

```
m14 <- glmer(count ~ pas + task + (pas|subject), data = data.count, family = poisson, control = glmerCon

# Finding sum of residual variance
tibble(sum(residuals(m13)^2),
       sum(residuals(m14)^2))
```

```
## # A tibble: 1 x 2
##   `sum(residuals(m13)^2)` `sum(residuals(m14)^2)`
##                     <dbl>                   <dbl>
## 1                   2508.                   2749.
```

```
# Comparing AIC
AIC(m13, m14)
```

```
##     df      AIC
## m13  9 4685.119
## m14  7 4923.190
```

When comparing the two models, I can conclude that the interaction-model (m13) both has lower sum of residual variance (SSR) and a lower AIC value, hence including the interaction makes my model substantially better.

v. indicate which of the two models, you would choose and why

If this was a post-hoc analysis I guess I would choose the model including the interaction as this clearly is a 'better' model. However, on a more conceptual note, I would depend on my theoretical background and knowledge. If there's evidence suggesting that the number of digits shown (expectations) in some way effect how clearly the stimuli is perceived, I would include the interaction. Or if I asked about the relation of the two as part of my research question.

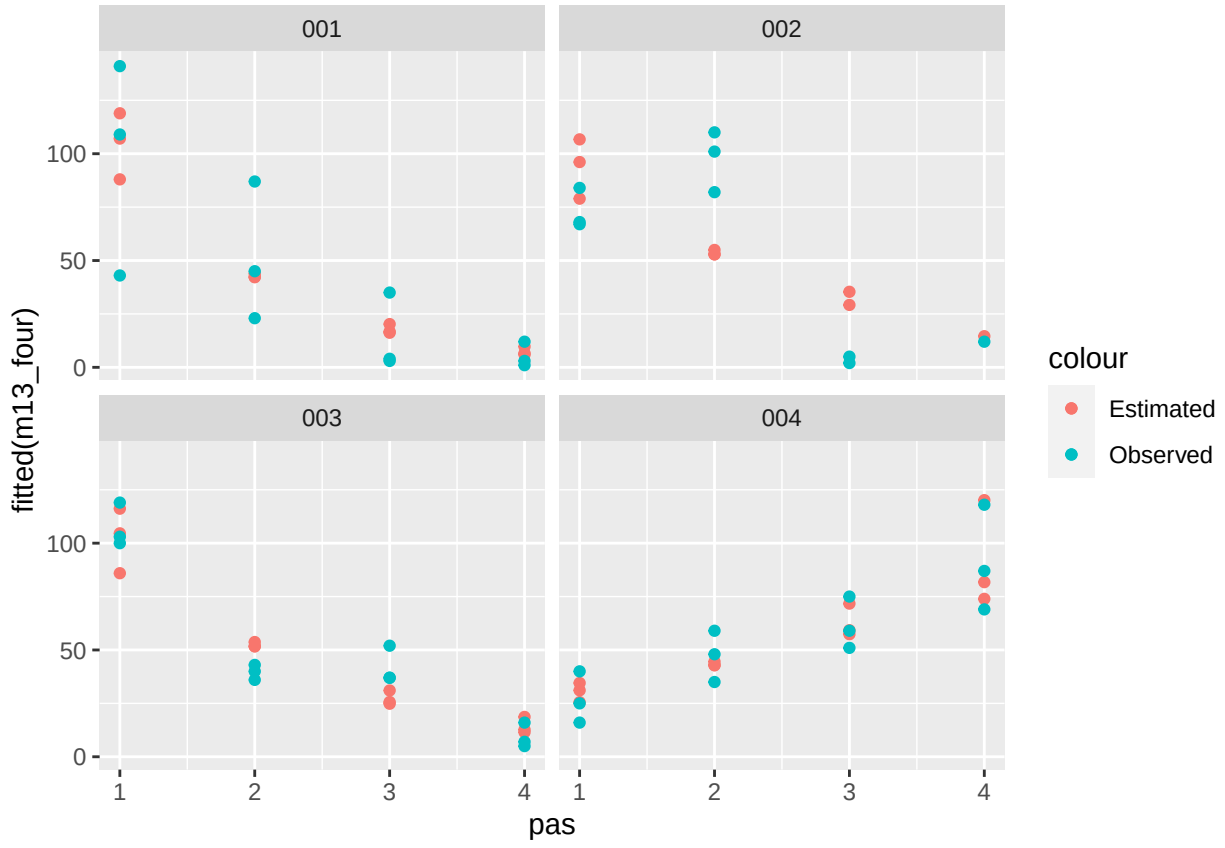
vi. based on your chosen model - write a short report on what this says about the distribution of ratin

vii. include a plot that shows the estimated amount of ratings for four subjects of your choosing

```
data.count_four <- data.count %>%
  filter(subject == "001" | subject == "002" | subject == "003" | subject == "004")

m13_four <- glmer(count ~ pas * task + (pas|subject), data = data.count_four, family = poisson)

data.count_four %>%
    ggplot() +
      geom_point(aes(x = pas, y = fitted(m13_four), color = "Estimated")) +
      geom_point(aes(x = pas, y = count, color = "Observed")) +
      facet_wrap( ~ subject)
```



3) Finally, fit a multilevel model that models *correct* as dependent on *task* with a unique intercept for each *subject*

```
m15 <- lme4::glmer(correct ~ task + (1|subject), data = df, family = binomial)
```

i. does _task_ explain performance?

Yes, it does indeed. In the quadruplets task the performance (correct) is significantly worse than the pairs-task, but the performance in single-task is better than pairs-task. Or in short: Task significantly predicts correctness for all task levels (all $p < 0.05$).

ii. add _pas_ as a main effect on top of _task_ - what are the consequences of that?

```
m16 <- lme4::glmer(correct ~ task + pas + (1|subject), data = df, family = binomial)
```

It makes the influence of task on performance irrelevant (insignificant.)

iii. now fit a multilevel model that models _correct_ as dependent on _pas_ with a unique intercept for

```
m17 <- lme4::glmer(correct ~ pas + (1|subject), data = df, family = binomial)
```

iv. finally, fit a model that models the interaction between _task_ and _pas_ and their main effects

```
m18 <-lme4::glmer(correct ~ pas * task + (1|subject), data = df, family = binomial)
```

v. describe in your words which model is the best in explaining the variance in accuracy

```
AIC(m15, m16, m17, m18)
```

```
##     df      AIC
## m15  4 19927.21
## m16  5 17425.01
## m17  3 17421.46
## m18  7 17422.82
```

Only model 15 and 17 have significance for all predictors. When comparing AIC values, I see that model 17 have a quite substantially lower value compared to model 15. This means that this model has the best balance between complexity and explanatory power.