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" (factor): indicate if the participant did the first experiment ('staircase') or the follow-up study ('experiment'). "pas" (factor): 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" (factor): 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" (factor): 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" (factor): 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)

```
df <- df %>%
  mutate(correct =  as.logical(correct)) %>%
  mutate(subject = as.factor(subject)) %>%
  mutate(task = as.factor(task)) %>%
  mutate(cue = as.factor(cue)) %>%
  mutate(pas = as.factor(pas)) %>%
  mutate(trial = as.factor(trial)) %>%
  mutate(trial.type = as.factor(trial.type))
```

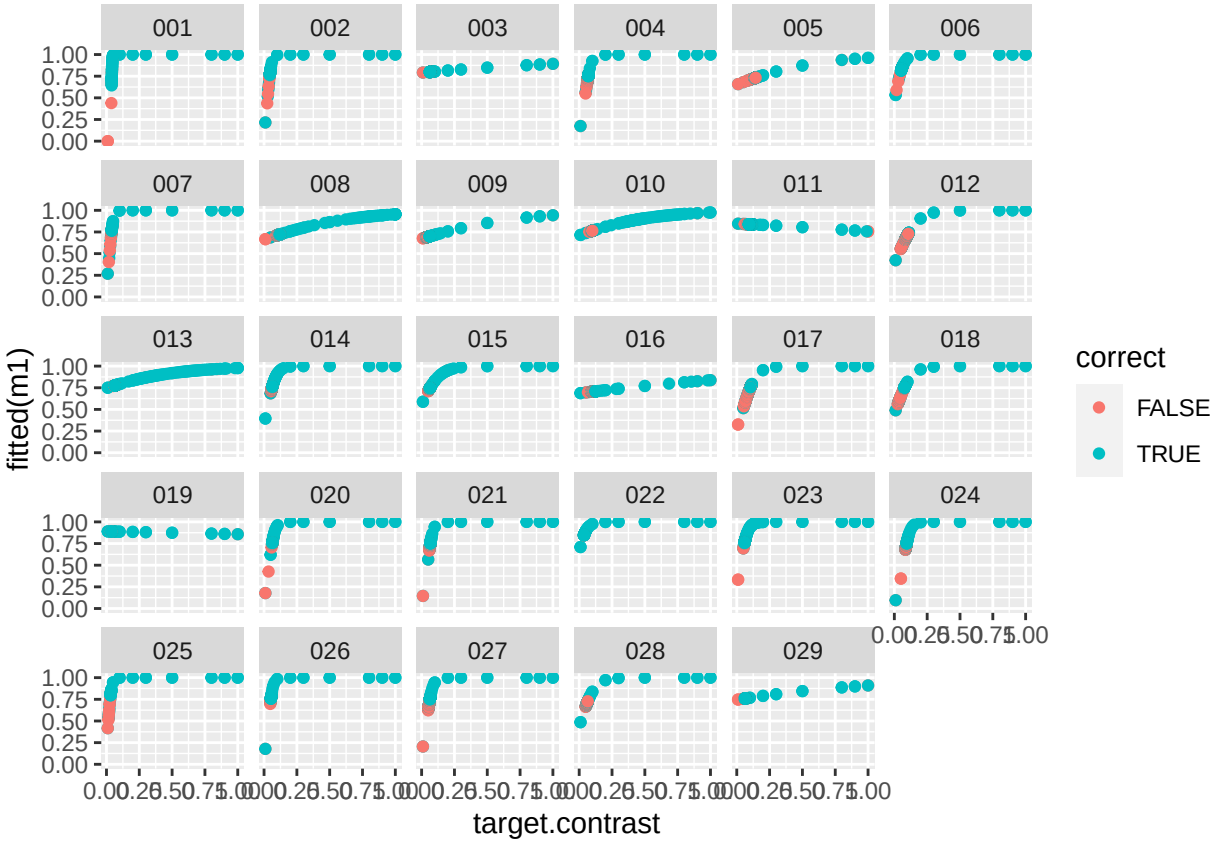
iii. for the staircasing part __only__, create a plot for each subject where you plot the estimated fun

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

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

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

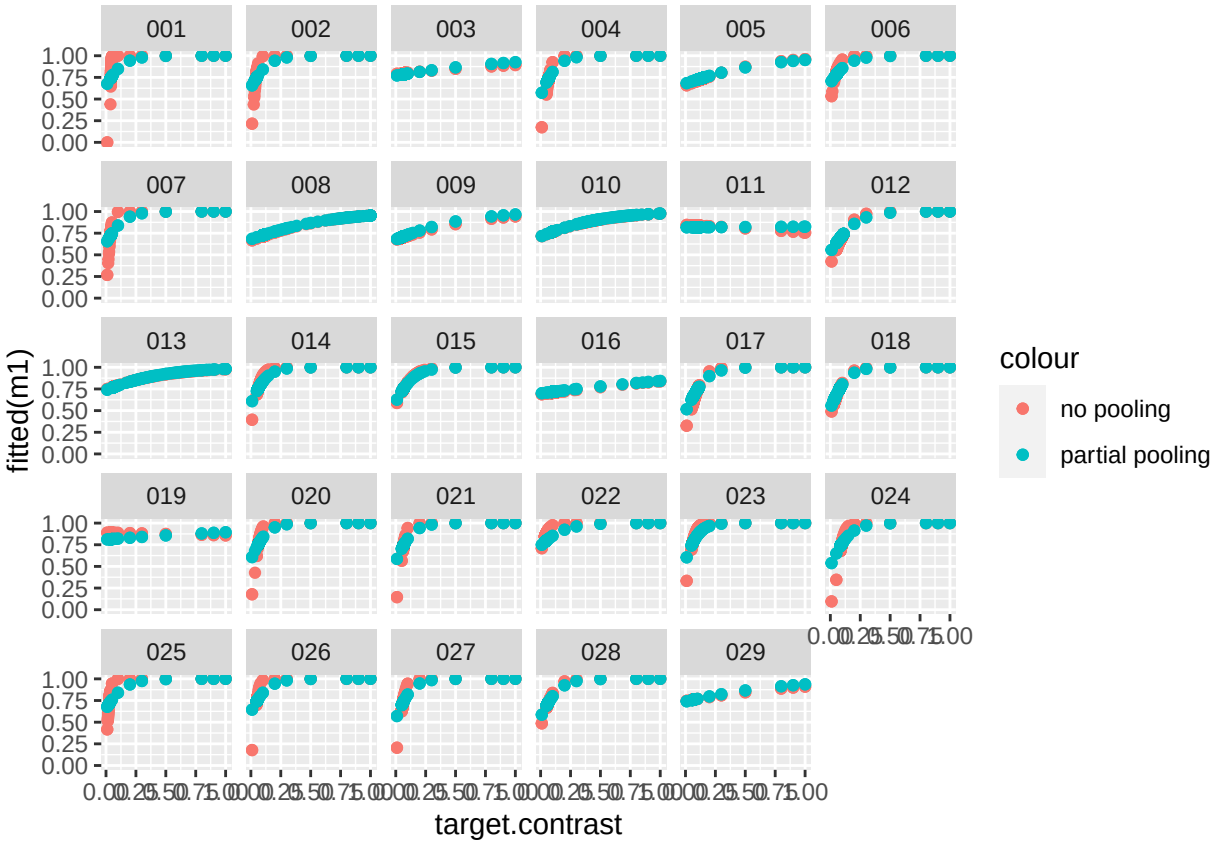
```
ggplot(data = df_staircase, aes(x = target.contrast, y = fitted(m1), color = correct)) +
  geom_point() +
  facet_wrap( ~ subject)
```



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 = "no 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. Therefore it's important to "tell" the model that there will be individual differences. However whilst the the 'no-pooling' model will be extremely good at describing our particualr data-set, it's not very generalisable, this is why the partial pooling model is to prefer.

Exercise 2

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

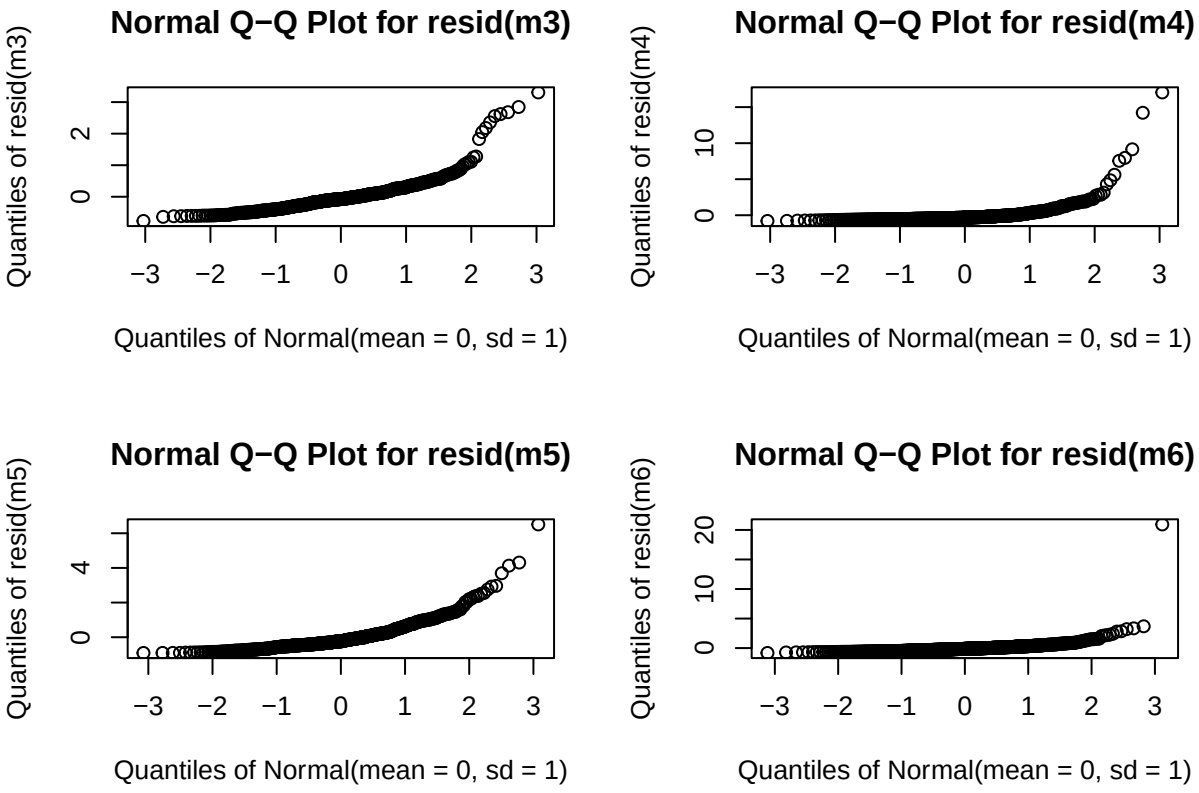
```
df_experiment <- df %>%
  filter(trial.type == "experiment")
```

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

```r
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"),]

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))
```



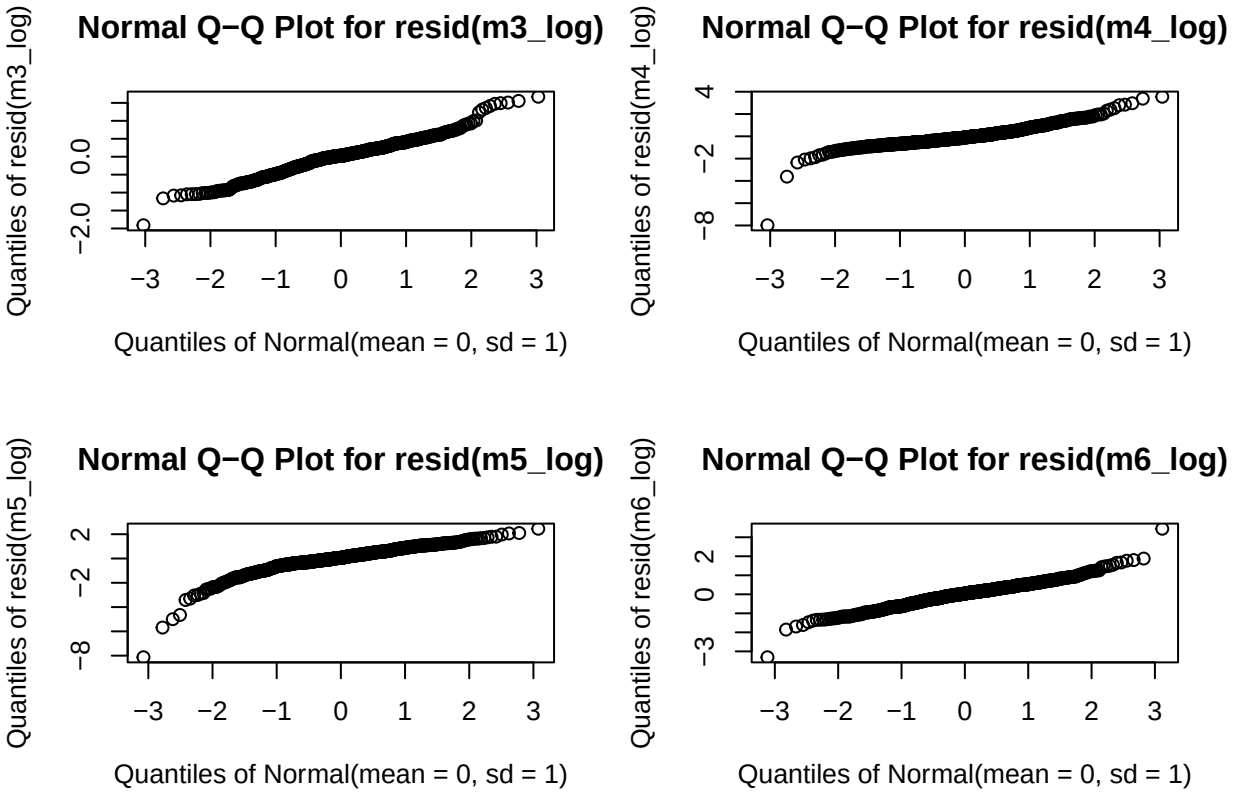
i. comment on these

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?

```
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)

```
# Making all sorts of models
m1_obj <- lmerTest::lmer(log(rt.obj) ~ task + (1|subject), data = df_experiment, REML = FALSE)
m2_obj <- lmerTest::lmer(log(rt.obj) ~ task + (1|subject) + (1|pas), data = df_experiment, REML = FALSE)
m3_obj <- lmerTest::lmer(log(rt.obj) ~ task + (1|subject) + (1|trial), data = df_experiment, REML = FALS
m4_obj <- lmerTest::lmer(log(rt.obj) ~ task + (1+task|subject), data = df_experiment, REML = FALSE)
```

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

```
## Warning: Model failed to converge with 1 negative eigenvalue: -7.6e+02
```

```
m5_obj <- lmerTest::lmer(log(rt.obj) ~ task + (1+task|trial)+ (1|subject), data = df_experiment, REML =
```

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

```
## Warning: Model failed to converge with 1 negative eigenvalue: -1.9e+03
```

```
m6_obj <- lmerTest::lmer(log(rt.obj) ~ task + (1+task|pas) + (1|subject), data = df_experiment, REML = 
```

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

```
## Warning: Model failed to converge with 1 negative eigenvalue: -1.4e+02
```

```
model <- c("m1_obj", "m2_obj", "m3_obj")
sigma <- c(sigma(m1_obj), sigma(m2_obj), sigma(m3_obj)) # finding residual standard deviation
AIC <- c(AIC(m1_obj), AIC(m2_obj), AIC(m3_obj)) # finding AIC values
as.tibble(cbind(model, sigma, AIC)) # making a data-frame to compare values easily
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
## # A tibble: 3 x 3
##   model  sigma             AIC
##   <chr>  <chr>             <chr>
## 1 m1_obj 0.786510836284884 29685.3126188651
## 2 m2_obj 0.76761263445794  29093.9491734722
## 3 m3_obj 0.767353686787788 29459.9382110624
```

Conceptually I would sure have "subject" as random intercept as I except different baseline levels pr. participant and I also guess that ignoring it would make *pseudo-replication*, therefore "subject" is included in all models.

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, therefore all models having this (m4_obj, m5_obj, m6_obj) were excluded early on.

m2_obj and m3_obj have the lowest residual standard deviation (sigma), which are nearly identical. However, m2_obj has the absolute lowest AIV value, therefore this would be my chosen model: m2_obj = log(rt.obj) ~ task + (1|subject) + (1|pas).

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

```
coef(summary(m2_obj))
```

```
##                   Estimate Std. Error          df   t value     Pr(>|t|)
## (Intercept)    -0.29543506 0.13400879     7.95183 -2.204595 5.877231e-02
## taskquadruplet -0.07863297 0.01680514 12496.19720 -4.679103 2.911560e-06
## tasksingles    -0.13698236 0.01690658 12499.23459 -8.102310 5.891496e-16
```

The estimates for quadruplet_task and singles_task are *negative* which tells me that the reaction time in these trials were faster than in the pairs_task.

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

```
m1_taskpas <- lmerTest::lmer(log(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
```

```
m2_taskpas <- lmerTest::lmer(log(rt.obj) ~ task * pas + (1|subject) + (1|task), data = df_experiment, RE
```

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

```
m3_taskpas <- lmerTest::lmer(log(rt.obj) ~ task * pas + (1|subject) + (1|pas), data = df_experiment, REM
```

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

```
m4_taskpas <- lmerTest::lmer(log(rt.obj) ~ task * pas + (1|subject) + (1|trial), data = df_experiment, R
m5_taskpas <- lmerTest::lmer(log(rt.obj) ~ task * pas + (1|subject) + (1|trial) + (1|cue), data = df_exp
m6_taskpas <- lmerTest::lmer(log(rt.obj) ~ task * pas + (1|subject) + (1|trial) + (1|cue) + (1|target.co
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unider
##  - Rescale variables?
```

Without getting an error I can add three random intercepts: m5_taskpas = log(rt.obj) ~ task * pas + (1|subject) + (1|trial) + (1|cue)

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

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

```
print(VarCorr(m3_taskpas), comp='Variance')
```

```
##  Groups   Name        Variance
##  subject  (Intercept) 0.15675
##  pas      (Intercept) 0.00000
##  Residual             0.58619
```

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

I see that adding 'pas' 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

```
m_count_int <- 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| = 0.00328229 (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?

Since pas is a factor ie. distinct separate levels, it doesn't really make sense to model a slope between the levels since a slope kinda assumes a continuous variable.

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

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

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

```
m_null_count <- glmer(count ~ 1 + (pas|subject), data = data.count, family = poisson)
m_count <- glmer(count ~ pas + task + (pas|subject), data = data.count, family = poisson, control = glm

text_2 <- c("m_null_count", "m_count", "m_count_int")
sigma_2 <- c(sigma(m_null_count), sigma(m_count), sigma(m_count_int))
AIC_2 <- AIC(m_null_count, m_count, m_count_int)
resid_var <- c(sum(residuals(m_null_count)^2), sum(residuals(m_count)^2), sum(residuals(m_count_int)^2)
as.tibble(cbind(text_2, resid_var, sigma_2, AIC_2))
```

```
## # A tibble: 3 x 5
##   text_2       resid_var sigma_2    df   AIC
##   <chr>            <dbl>   <dbl> <dbl> <dbl>
## 1 m_null_count      962.       1    11 3397.
## 2 m_count           962.       1    16 3399.
## 3 m_count_int       699.       1    22 3148.
```

When comparing the two models to the baseline model (m_null_count) I see that the model that includes the interaction has the lowest AIC value. In fact the model *not* including the interaction has an AIC and residual variance similar to the baseline model, indicating that this is not a very good model. The interaction model also has the lowest residual variance, supporting that this model is best.

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

I would choose the model including the interaction as this clearly is a 'better' model. I would thousg consider that adding the interaction also adds a substantial amount of parameters (see df) hence making the model very complex. I would though trust that since I didn't get a warning or error when creating the model, it's within an alright limit.

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

```
coef(summary(m_count_int))
```

```
##                         Estimate Std. Error    z value      Pr(>|z|)
## (Intercept)           4.0357061 0.10975695 36.7694817 5.677992e-296
## pas2                 -0.0237800 0.11962544 -0.1987872  8.424292e-01
## pas3                 -0.5136565 0.20717492 -2.4793372  1.316268e-02
## pas4                 -0.7729351 0.29075037 -2.6584149  7.850918e-03
## taskquadruplet        0.1148999 0.03127246  3.6741564  2.386365e-04
## tasksingles          -0.2309462 0.03418206 -6.7563571  1.415044e-11
## pas2:taskquadruplet -0.1137545 0.04605306 -2.4700758  1.350844e-02
## pas3:taskquadruplet -0.2090142 0.05286545 -3.9537009  7.695158e-05
## pas4:taskquadruplet -0.2150035 0.05229939 -4.1110131  3.939268e-05
## pas2:tasksingles      0.1953636 0.04829800  4.0449636  5.233126e-05
## pas3:tasksingles      0.2429945 0.05369198  4.5257140  6.019194e-06
## pas4:tasksingles      0.5634628 0.05101389 11.0452833  2.310376e-28
```

Main effects We see that "Pas2" will reduce count frequency compared to "Pas1" (baseline). Generally every time you go a step up in "Pas" you will reduce count frequency even further.

We see that going from pairs_task (baseline) to quadruplets_task will increase count-frequency whilst going from pairs_task (baseline) to singles_task will decrease it.

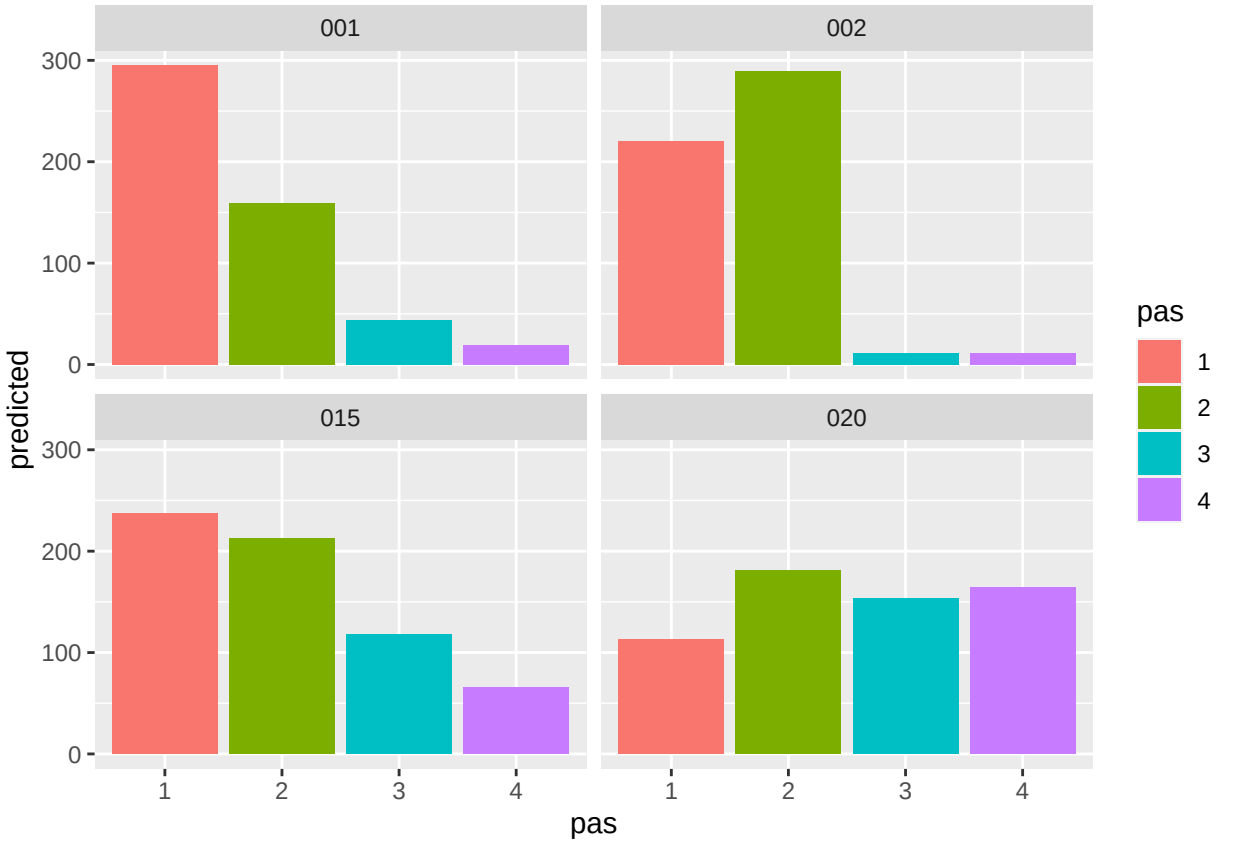
Interaction Effects We see that quadruplet interactions with all pas levels (pas2, pas3, pas4) will *decrease* the count frequency which singles interactions with all pas levels will *increase* our count frequency.

`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 == "015" | subject == "020")

data.count_four$predicted <- exp(predict(m_count_int, newdata = data.count_four)) # making a column giv

ggplot(data.count_four, aes(x = pas, y = predicted, fill = pas)) +
  geom_bar(stat = 'identity') +
  facet_wrap(~ subject)
```



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

```
m1_correct <- lme4::glmer(correct ~ task + (1|subject), data = df, family = binomial)
coef(summary(m1_correct))
```

```
##                   Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)     1.10071177 0.08386364 13.125018 2.367280e-39
## taskquadruplet -0.09825083 0.04189262 -2.345302 1.901168e-02
## tasksingles     0.18541782 0.04335937  4.276303 1.900223e-05
```

```
i. does _task_ explain performance?
```

Yes, it does indeed. In the quadruplets task the performance (correct) is significantly worse than the pairs-task (baseline), but the performance in single-task is better than pairs-task (baseline). 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?
```

```
m2_correct <- lme4::glmer(correct ~ task + pas + (1|subject), data = df, family = binomial)
```

It makes the influence of "task" on performance irrelevant (insignificant) suggesting that "pas" is a better predictor for performance accuracy (correct).

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

```
m3_correct <- 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

```
m4_correct <-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

```
m_null_correct <- lme4::glmer(correct ~ 1 + (1|subject), data = df, family = binomial) # creating a nul

text_3 <- c("m_null_correct", "m1_correct", "m2_correct", "m3_correct", "m4_correct")
AIC_3 <- AIC(m_null_correct, m1_correct, m2_correct, m3_correct, m4_correct)
sigma_3 <- c(sigma(m_null_correct), sigma(m1_correct), sigma(m2_correct), sigma(m3_correct), sigma(m4_c
resid_var_3 <- c(sum(residuals(m_null_correct)^2), sum(residuals(m1_correct)^2), sum(residuals(m2_corre


as.tibble(cbind(text_3, AIC_3, sigma_3, resid_var_3))
```

```
## # A tibble: 5 x 5
##   text_3            df    AIC sigma_3 resid_var_3
##   <chr>          <dbl>  <dbl>   <dbl>       <dbl>
## 1 m_null_correct     2 19968.       1      19849.
## 2 m1_correct         4 19927.       1      19804.
## 3 m2_correct         7 17425.       1      17297.
## 4 m3_correct         5 17421.       1      17297.
## 5 m4_correct        13 17431.       1      17291.
```

Solely based on AIC values m3_correct (correct ~ pas + (1|subject)) is the best model. I do though see that the sum of residual variance (SSR) is lowest for the model that includes the interaction between pas and task m4_correct (correct ~ pas * task + (1|subject)). However, this model has a lot more parameters (df) and whilst it does lower the SSR, it's not by very much. I would argue that extra variance explained does not outweigh the more complexity you add. Therefore I would choose m3_correct (correct ~ pas + (1|subject)).