Portfolio 2.1, Methods 3, 2021, autumn semester

Laura W. Paaby

4/9 - 2021

```
#loading libraries:
pacman::p_load(tidyverse, readbulk, patchwork, lmerTest, ggpubr, dfoptim)
```

added new line

Exercises and objectives

The objectives of the exercises of this assignment are:
1) Download and organise the data and model and plot staircase responses based on fits of logistic functions
2) Fit multilevel models for response times
3) Fit multilevel models for count data

REMEMBER: In your report, make sure to include code that can reproduce the answers requested in the exercises below (**MAKE A KNITTED VERSION**)
REMEMBER: This assignment will be part of your final portfolio

Exercise 1

Go to https://osf.io/ecxsj/files/ (https://osf.io/ecxsj/files/) and download the files associated with Experiment 2 (there should be 29).
The data is associated with Experiment 2 of the article at the following DOI https://doi.org/10.1016/j.concog.2019.03.007 (https://doi.org/10.1016/j.concog.2019.03.007)

1. Put the data from all subjects into a single data frame
2. Describe the data and construct extra variables from the existing variables
    i. add a variable to the data frame and call it *correct* (have it be a *logical* variable). Assign a 1 to each row where the subject indicated the correct answer and a 0 to each row where the subject indicated the incorrect answer (**Hint:** the variable *obj.resp* indicates whether the subject answered “even”, *e* or “odd”, *o*, and the variable *target_type* indicates what was actually presented.

```
#the target.type and the obj.response should match
data$correct <- ifelse((data$obj.resp == "o" & data$target.type =="odd") | (data$obj.
resp == "e" & data$target.type =="even"), 1,0)

#skimming the data to see the classes of the variables
ls.str(data)
```

```
## correct :  num [1:18131] 1 1 1 1 1 1 1 1 1 1 ...
## cue :  num [1:18131] 29 29 29 29 29 29 29 29 29 29 ...
## even.digit :  num [1:18131] 8 4 4 4 4 4 4 4 4 8 ...
## File :  chr [1:18131] "001.csv" "001.csv" "001.csv" "001.csv" "001.csv" "001.csv"
...
## jitter.x :  num [1:18131] -0.3425 0.0623 -0.4058 -0.3615 0.2891 ...
## jitter.y :  num [1:18131] 0.4489 0.0291 0.4995 -0.2224 0.4132 ...
## obj.resp :  chr [1:18131] "o" "e" "e" "o" "e" "e" "o" "o" "e" "o" "o" "e" "o" "o"
"o" ...
## odd.digit :  num [1:18131] 9 9 7 7 7 7 7 7 9 9 ...
## pas :  num [1:18131] 4 4 4 4 4 4 4 4 4 4 ...
## rt.obj :  num [1:18131] 1.158 2.456 0.951 0.62 0.95 ...
## rt.subj :  num [1:18131] 1.754 1.39 0.686 0.653 0.582 ...
## seed :  num [1:18131] 9817 9817 9817 9817 9817 ...
## subject :  chr [1:18131] "001" "001" "001" "001" "001" "001" "001" "001" "001" "00
1" ...
## target.contrast :  num [1:18131] 1 1 0.9 0.9 0.8 0.8 0.5 0.5 0.3 0.3 ...
## target.frames :  num [1:18131] 3 3 3 3 3 3 3 3 3 3 ...
## target.type :  chr [1:18131] "odd" "even" "even" "odd" "even" "even" "odd" "odd"
"even" ...
## task :  chr [1:18131] "pairs" "pairs" "pairs" "pairs" "pairs" "pairs" "pairs" ...
## trial :  num [1:18131] 0 1 2 3 4 5 6 7 8 9 ...
## trial.type :  chr [1:18131] "staircase" "staircase" "staircase" "staircase" "stair
case" ...
```

```
ii. describe what the following variables in the data frame contain, _trial.type_, _p
as_, _trial_, _target.contrast_, _cue_, _task_, _target_type_, _rt.subj_, _rt.obj_, _
obj.resp_, _subject_ and _correct_. (That means you can ignore the rest of the variab
les in your description). For each of them, indicate and argue for what `class` they
should be classified into, e.g. _factor_, _numeric_ etc.
```

TRIAL TYPE: *trial type is at the moment a character, but should be a factor, so that the performance can be modeled and compared by which trial was done*

```
#trial_type:
data$trial.type <- as.factor(data$trial.type)
```

PAS: *is a subjective rating and indicates whether you have seen the stimulus or not: 1 (no experience), 2 (weak glimpse), 3 (almost clear experience), 4 (clear experience). It is measures on the Perceptual Awareness Scale, and should be a factor since it is a categorical ordered variable.*

```
#pas:
data$pas <- as.factor(data$pas)
```

TARGET CONTRAST: *is the contrast in the stimulus shown relevant to the background and adjusted to each participants threshold. It is numeric, and should stay in that way since*

```
#target.contrast:
data$target.contrast <- as.numeric(data$target.contrast)
```

TRIAL: *number of trial, which reset as the experiment begins. It is nominal data, and should be encoded as a factor.*

```
#trial:
data$trial <- as.factor(data$trial)
```

CUE: *is 36 different combinations of numbers participants can be cued with, 3 types depending on the task setting. Again nominal, so factor.*

```
#cue:
data$cue <- as.factor(data$cue)
```

TASK: *is a character and should be a factor, since it is nominal. It is the task setting, and can be either singles, paired or quadruplets referring to the amount of letters showing in the cue.*

```
#task:
data$task <- as.factor(data$task)
```

TARGET TYPE: *Indicates whether the target was an even or odd number. It is binary data, and should be coded as a factor.*

```
#target_type:
data$target.type <- as.factor(data$target.type)
```

RT.SUBJ: *reaction time of the rating of the pass - their confidence in how clear they saw the target. This is numeric data and should stay that way, since we are dealing with reacting time, which here is a continuous variable.*

RT.OBJ: *reaction time of the answer on the task. Same as rt.subj*

OBJ.RESP: *the subject's to if the letter is odd or even, thus binomial and should be a factor*

```
#obj.resp:
data$obj.resp <- as.factor(data$obj.resp)
```

SUBJECT: *Participant index and should be a factor*

```
#subject:
data$subject <- as.factor(data$subject)
```

CORRECT: *indicates whether the subject answered correct 1 or wrong 0 when answering whether the letter was odd or even. It is binomial, thus it should be a factor.*

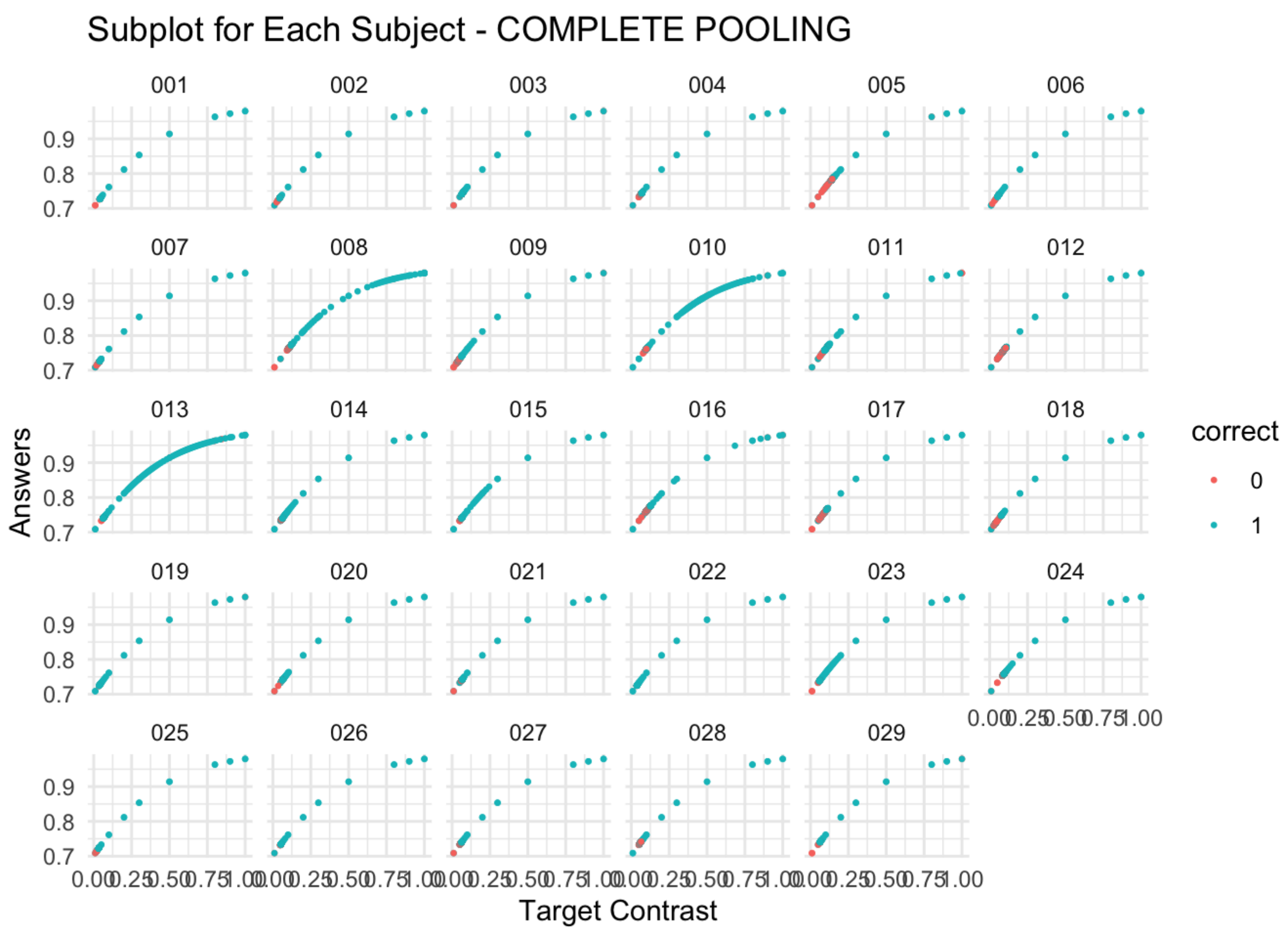
```
#correct:
data$correct <- as.factor(data$correct)
```

```
iii. for the staircasing part __only__, create a plot for each subject where you plot
the estimated function (on the _target.contrast_ range from 0-1) based on the fitted
values of a model (use `glm`) that models _correct_ as dependent on _target.contrast
_. These plots will be our _no-pooling_ model. Comment on the fits - do we have enoug
h data to plot the logistic functions?
```

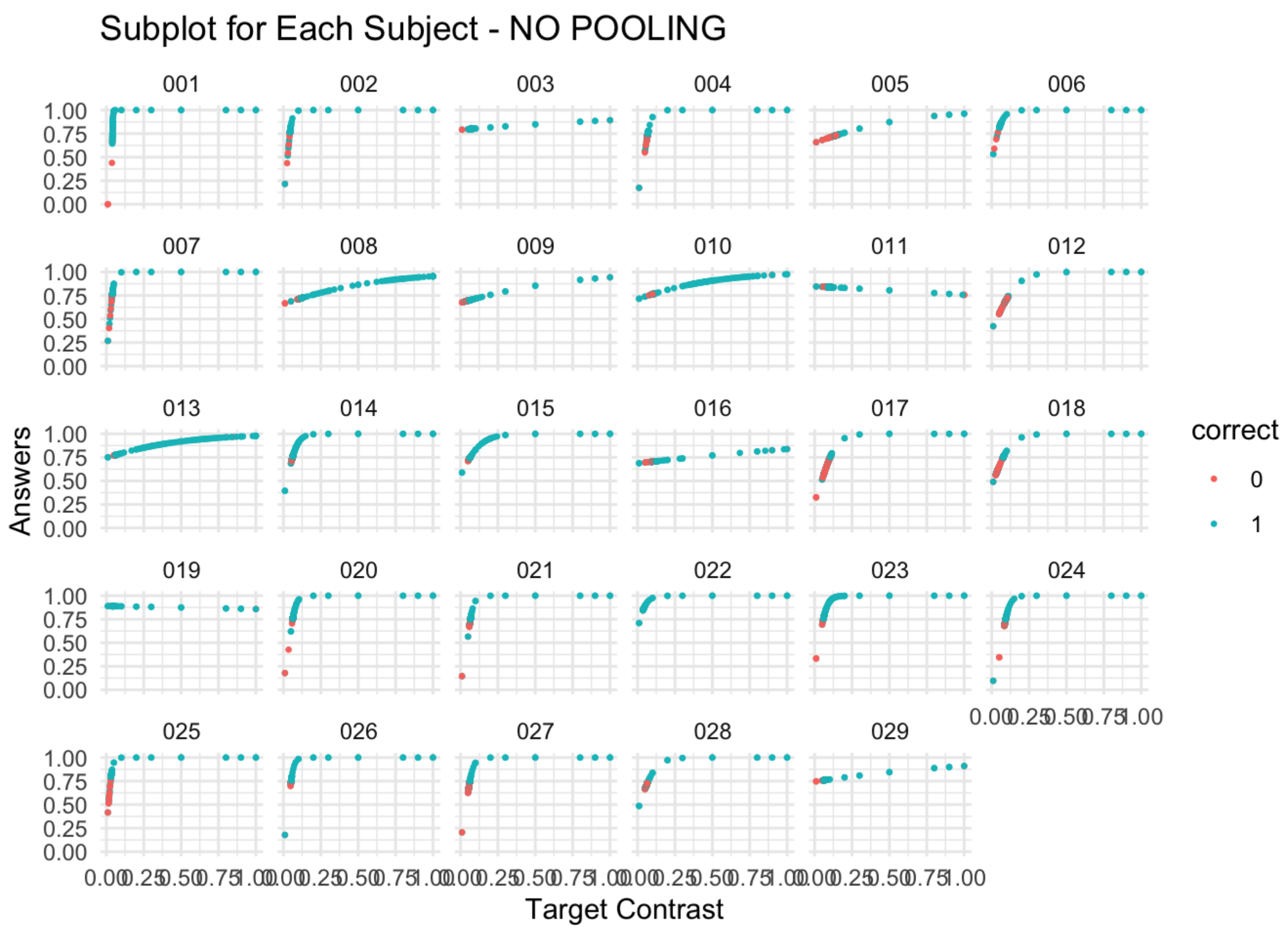
```
df_m <- data %>%
  filter(trial.type == "staircase")


#model for COMPLETE pooling!!!!
trial_model<- glm(correct ~ target.contrast, data = df_m, family = "binomial")
fitted_y <- fitted(trial_model)

plot1 <- ggplot(df_m, aes(target.contrast, fitted_y, color = correct)) +
    geom_point(size =0.5)+
    facet_wrap(~subject) +
    theme_minimal()+
    xlab("Target Contrast")+
    ylab('Answers') +
    ggtitle("Subplot for Each Subject - COMPLETE POOLING")




#model for NO pooling:
##### meaning that we only take into account the individual and does not pool the dat
a together.
no.pool <- glm(correct ~ target.contrast + subject + subject:target.contrast, data =
 df_m, family = "binomial" )
```

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

```
fit_np <- fitted(no.pool)

plot2 <- ggplot(df_m, aes(target.contrast, fit_np, color = correct)) +
    geom_point(size = 0.5)+
    facet_wrap(~subject) +
    theme_minimal()+
    xlab("Target Contrast")+
    ylab('Answers') +
    ggtitle("Subplot for Each Subject - NO POOLING")

plot1
```

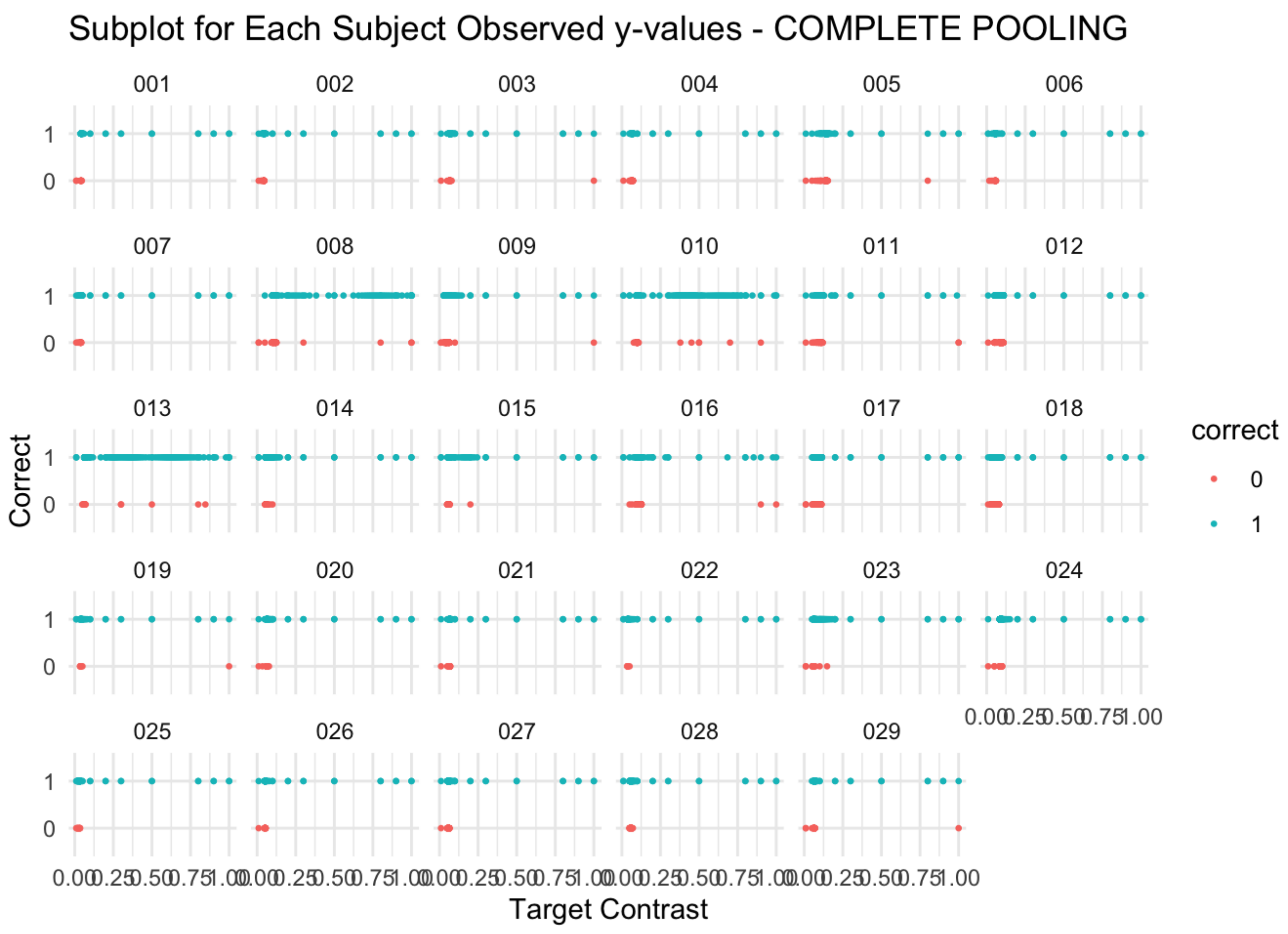


```
plot2
```



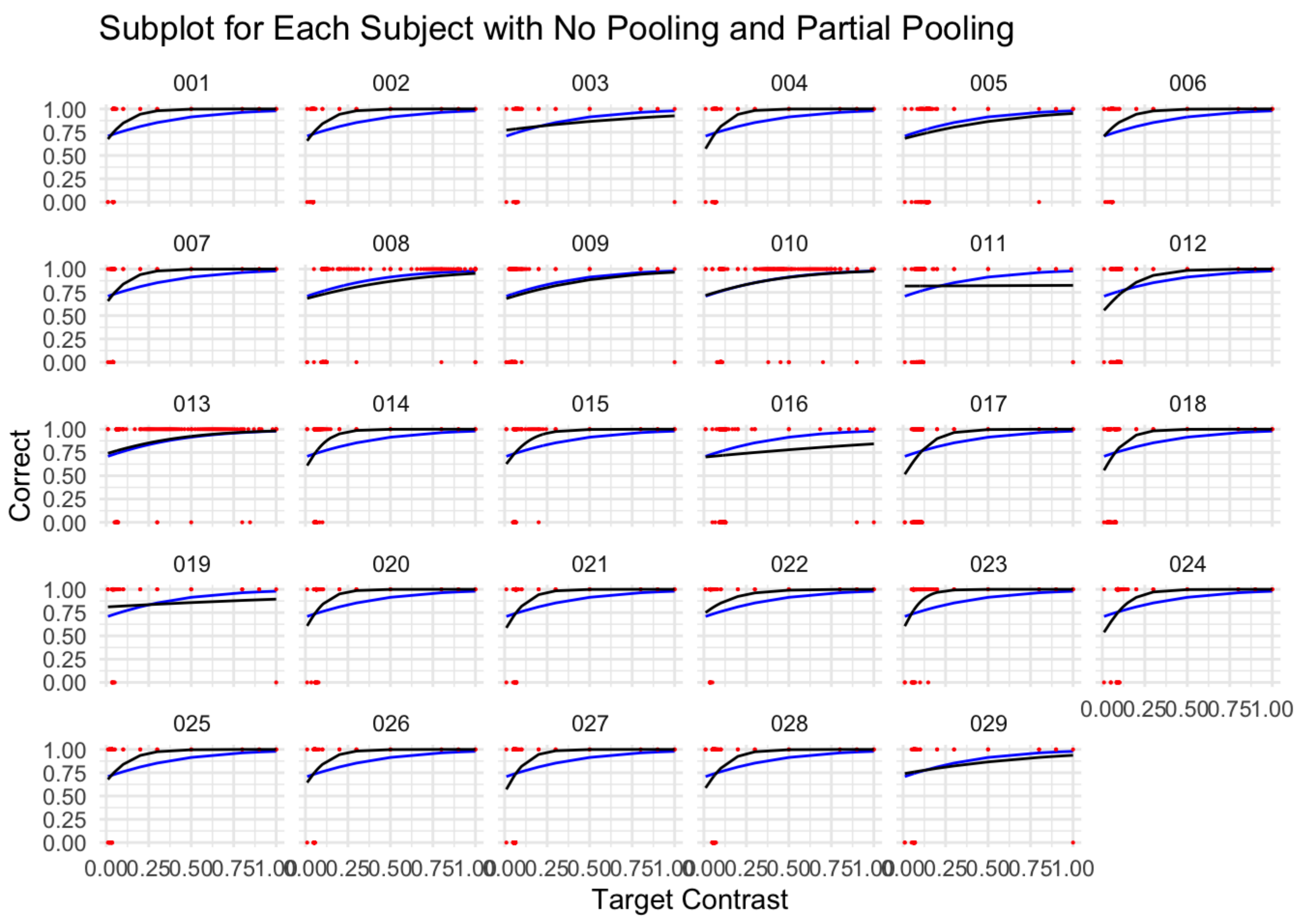
These plots are both supposed to follow a sigmoid function, assuming the relation is logistic, but this is not really the case. There appears to be a ceiling effect, since almost all subjects had way more correct answers than wrong, meaning almost all of the points are at y = 1. However, compared to the complete pooling plot, no pooling is a better fit for each subject. Moreover, it appears as if more data is needed to succesfully plot the logistic regression.

```
#complete pooling with observed y-values.
plot3 <- ggplot(df_m, aes(target.contrast, correct, color = correct)) +
    geom_point(size =0.5)+
    facet_wrap(~subject) +
    theme_minimal()+
    xlab("Target Contrast")+
    ylab('Correct') +
    ggtitle("Subplot for Each Subject Observed y-values - COMPLETE POOLING")

plot3
```



```
iv. on top of those plots, add the estimated functions (on the _target.contrast_ rang
e from 0-1) for each subject based on partial pooling model (use `glmer` from the pac
kage `lme4`) where unique intercepts and slopes for _target.contrast_ are modelled fo
r each _subject_
```

```
#partial pooled model:
library(lme4)
part.pool.model <- glmer(correct ~ target.contrast + (target.contrast|subject), famil
y = "binomial", data = df_m)

part_fit <- fitted(part.pool.model)



plot3 <- ggplot(df_m, aes(target.contrast, fitted_y)) +
    geom_line(colour = 'blue', size = 0.5)+
    geom_point(aes(x = target.contrast, y = as.numeric(as.character(correct))), colou
r = 'red', size = 0.07)+
    geom_line(aes(y=part_fit), colour = 'black', size = 0.5) +
    facet_wrap(~subject) +
    theme_minimal()+
    xlab("Target Contrast")+
    ylab('Correct') +
    ggtitle("Subplot for Each Subject with No Pooling and Partial Pooling")

plot3
```



the black line are here the partial pooling and the blue the no-pooling. The red points are the observed data points.

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

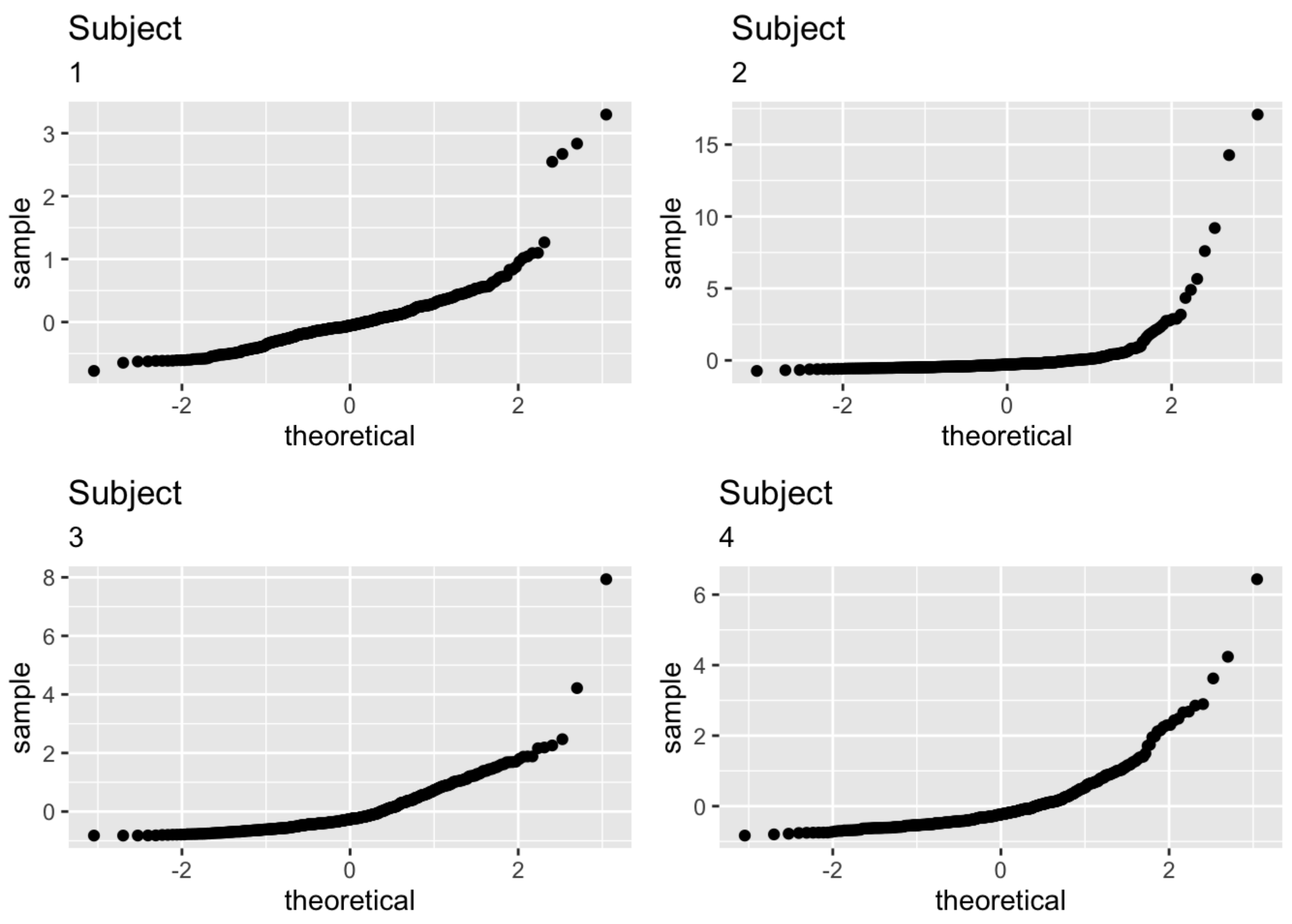
When partial pooling we are able to take into account both the average and each level of the categorical predictor. Hereby the model are more generalizable. The partial pooled points also appears to follow a sigmoid function a bit better than the not-pooled. It appears how the no pooled line actually fit the datapoints better, since it models different baselines and slopes for the subjects. However, this is a very hard model to generalise, thus the partial pooling model must be said to allow fot the better fit.

Exercise 2

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

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
#making a dataframe only containing the experimental trials.
experi.df <- data %>% filter(trial.type == "experiment")

df_1 <- experi.df %>%
    filter(subject == "001")
df_2 <- experi.df %>%
    filter(subject == "002")
df_3 <- experi.df %>%
    filter(subject == "003")
df_4 <- experi.df %>%
    filter(subject == "004")

#making the models for each subject's response time, with only the intercept modeled
m1 <- lm(rt.obj ~ 1, data = df_1)
m2 <- lm(rt.obj ~ 1, data = df_2)
m3 <- lm(rt.obj ~ 1, data = df_3)
m4 <- lm(rt.obj ~ 1, data = df_4)


#a function that plots a qq plot for the residuals of the given model
qq_f <- function(data.i, model.i, subject.i){
  plot <- data.i %>%
    ggplot(aes(sample = resid(model.i)))+
    geom_point(stat ="qq")+
    ggtitle("Subject", as.character(subject.i))

}

q1 <- qq_f(df_1, m1, 001)
q2 <- qq_f(df_2, m2, 002)
q3 <- qq_f(df_3, m3, 003)
q4 <- qq_f(df_4, m4, 004)

ggarrange(q1, q2, q3, q4)
```



i.

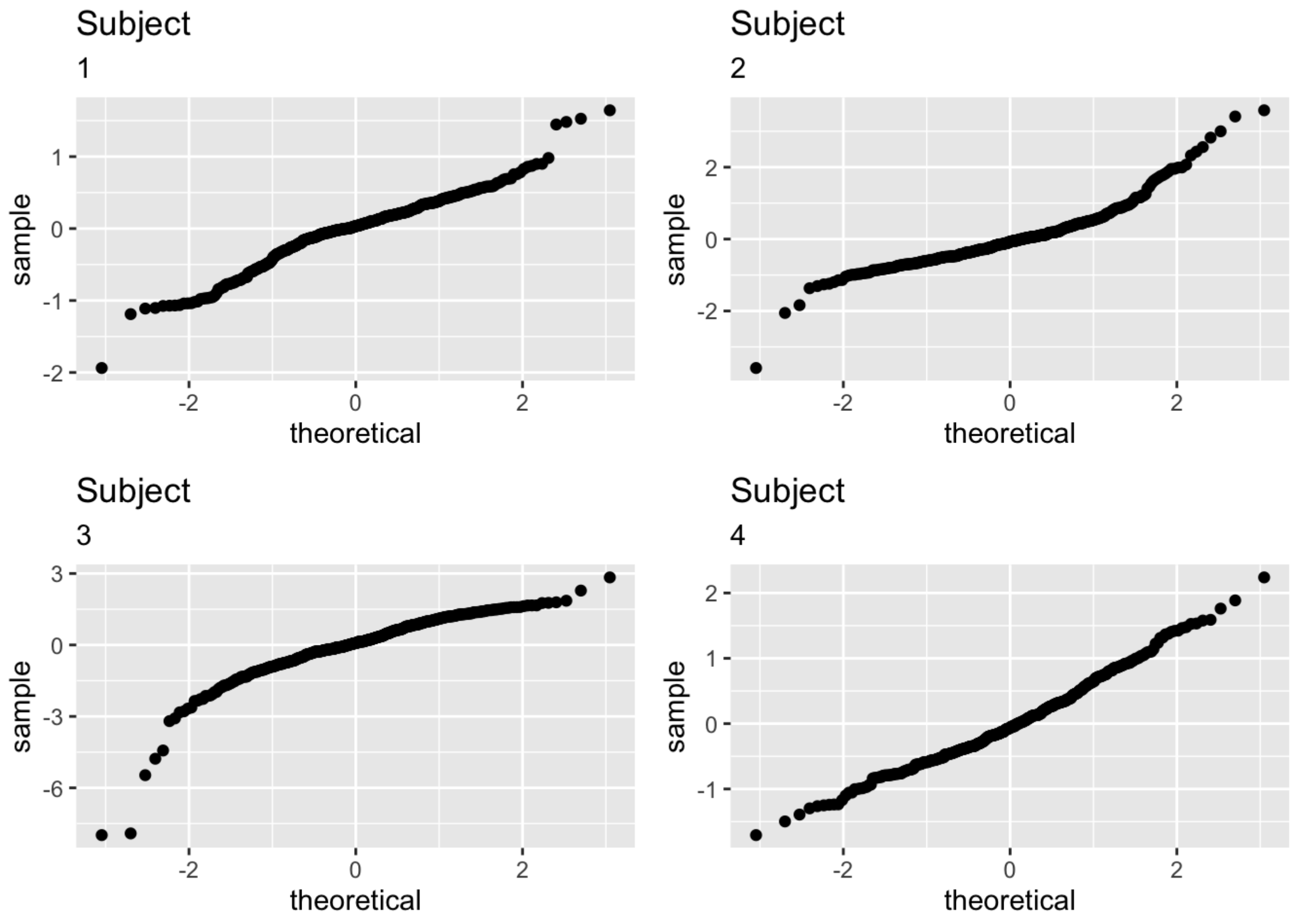
comment on these

It appears as if the values needs a transformation, since it looks rather skewed and doesn't look normally distributed.

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

make the exact same thing, but with log around the rt.obj in each model

```
#making the logged models
m1_log <- lm(log(rt.obj) ~ 1, data = df_1)
m2_log <- lm(log(rt.obj) ~ 1, data = df_2)
m3_log <- lm(log(rt.obj) ~ 1, data = df_3)
m4_log <- lm(log(rt.obj) ~ 1, data = df_4)

#using the functions to plot the qqplot over residuals now over the logged models
p1log <- qq_f(df_1, m1_log, 001)
p2log <- qq_f(df_2, m2_log, 002)
p3log <- qq_f(df_3, m3_log, 003)
p4log <- qq_f(df_4, m4_log, 004)

ggarrange(p1log, p2log, p3log, p4log)
```



2. Now do a partial pooling model modelling objective response times as dependent on *task*? (set `REML=FALSE` in your `lmer`-specification)
   this is done to the entire data set - I'm not sure however if what it asked is to only make the model on the four chosen participants. A sample size that small just doesn't seem that generalizable to me, so I went with all the participants.

   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) *Both subject (explained variance of 0.1241) and trial (explained variance of 0.1308) are chosen as random intercepts, since I'd argue that each subject must have a uniqe baseline when entering the task. This could be both their mental state, memory, IQ etc., which means that they should be compared to their own performance between tasks to take such factors into account, as well as the average across the group. I have chosen to make trials a random intercept as well, since the type and difficulty of trial might be affecting the performance. Looking at the summary, I find that despite these arguments, the variance for trial and subject is low compared to the variance explained by residuals (6.4924) ….. see the summary plot:*

```
#partial pooling model:
part_model <- lmer(rt.obj ~ task + (1|subject) + (1|trial), data = data, REML = FALSE
)
summary(part_model)
```

```
## Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
##   method [lmerModLmerTest]
## Formula: rt.obj ~ task + (1 | subject) + (1 | trial)
##    Data: data
##
##      AIC      BIC   logLik deviance df.resid
##  85716.7  85763.5 -42852.3  85704.7    18125
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
##  -1.859  -0.199  -0.088   0.073 112.414
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  trial    (Intercept) 0.1308   0.3617
##  subject  (Intercept) 0.1241   0.3523
##  Residual             6.4924   2.5480
## Number of obs: 18131, groups:  trial, 432; subject, 29
##
## Fixed effects:
##                  Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)     1.233e+00  7.544e-02  4.350e+01  16.344  < 2e-16 ***
## taskquadruplet -9.607e-02  4.666e-02  1.803e+04  -2.059   0.0395 *
## tasksingles    -1.900e-01  4.692e-02  1.810e+04  -4.049 5.16e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr
## taskqudrplt -0.310
## tasksingles -0.311  0.500
```

```
MuMIn::r.squaredGLMM(part_model)
```

```
## Warning: 'r.squaredGLMM' now calculates a revised statistic. See the help page.
```

```
##               R2m        R2c
## [1,] 0.0008899448 0.03864005
```

```
ii. explain in your own words what your chosen models says about response times betwe
en the different tasks
```

A significant difference in reaction time across all three types of tasks appears p <.05. i.e. when the cue changes (can be either singles, pairs or quadruplets). Going from pairs to single or quadruplets, the reaction times decreases significantly. The

R^2 is small, and not much of the variance is explained. Likewise, none of the chosen intercepts explains that much of the variance either: subject = 0.12 and trial = 0.13, compared to the residual variance 6.49

3. Now add *pas* and its interaction with *task* to the fixed effects
    i. how many types of group intercepts (random effects) can you add without ending up with convergence issues or singular fits?

```
inter_model <- lmer(rt.obj ~ pas*task + (1|subject) + (1 | trial) + (1|task) + (1|pa
s), data = data, REML = FALSE)
```

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

```
inter_model2 <- lmer(rt.obj ~ pas*task + (1|subject) + (1|trial) + (1|pas), data = da
ta, REML = FALSE)
```

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

```
inter_model3 <- lmer(rt.obj ~ pas*task + (1|subject) + (1|trial), data = data, REML =
FALSE)
```

the latter model is the only one of these I can make without running into convergence issues.

```
ii. create a model by adding random intercepts (without modelling slopes) that result
s in a singular fit - then use `print(VarCorr(<your.model>), comp='Variance')` to ins
pect the variance vector - explain why the fit is singular (Hint: read the first para
graph under details in the help for `isSingular`)
```

```
?isSingular
m_sing <- lmer(rt.obj ~ pas*task + (1|subject) + (1|trial) + (1|task), data = data, R
EML = FALSE)
```

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

*This gives the message:*vboundary (singular) fit: see ?isSingular *meaning that this model now results in a singular fit due to the random intercepts added.*

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

```
##  Groups   Name        Variance
##  trial    (Intercept) 0.13127
##  subject  (Intercept) 0.11273
##  task     (Intercept) 0.00000
##  Residual             6.46032
```

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

I assume the singularity occurs as an effect of overfitted data, which occurs when a model is too complex, i.e. it has too many predictors - both random intercept and slopes.

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 <- data %>%
  group_by(subject, pas, task) %>%
  summarise(count = n()) #using n to give the current size of the group, which gives
 the number of times eaxh subject categorized their experience as pas for each task,
 since these are the one called in group by.
```

```
## `summarise()` regrouping output by 'subject', 'pas' (override with `.groups` argum
ent)
```

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

```
#count modelled with slope for pas and subject and an interaction between pas and tas
k.
m_count <- glmer(count ~ pas*task + (1+pas|subject), family = poisson, data = data.co
unt)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00273537 (tol = 0.002, component 1)
```

it says that my model fails to converge…

```
i. which family should be used?
```

since we work with count data a Poisson distribution should be used

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

Since it is a factor, so we can’t make a general slope as we know them from continuous variables

```
iii. if you get a convergence error, try another algorithm (the default is the _Nelde
r_Mead_) - try (_bobyqa_) for which the `dfoptim` package is needed. In `glmer`, you
can add the following for the `control` argument: `glmerControl(optimizer="bobyqa")`
(if you are interested, also have a look at the function `allFit`)
```

```
m_count_new <- glmer(count ~ pas*task + (1+pas|subject), family = poisson, data = dat
a.count, glmerControl(optimizer="bobyqa"))

summary(m_count_new)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: count ~ pas * task + (1 + pas | subject)
##    Data: data.count
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##   3148.4   3232.7  -1552.2   3104.4      318
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.3871 -0.7853 -0.0469  0.7550  6.5438
##
## Random effects:
##  Groups  Name        Variance Std.Dev. Corr
##  subject (Intercept) 0.3324   0.5765
##          pas2        0.3803   0.6167   -0.75
##          pas3        1.1960   1.0936   -0.84  0.63
##          pas4        2.3736   1.5407   -0.86  0.42  0.72
## Number of obs: 340, groups:  subject, 29
##
## Fixed effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)           4.03570    0.10976  36.769  < 2e-16 ***
## pas2                 -0.02378    0.11963  -0.199 0.842461
## pas3                 -0.51365    0.20718  -2.479 0.013165 *
## pas4                 -0.77292    0.29076  -2.658 0.007853 **
## taskquadruplet        0.11490    0.03127   3.674 0.000239 ***
## tasksingles          -0.23095    0.03418  -6.756 1.42e-11 ***
## pas2:taskquadruplet -0.11376    0.04605  -2.470 0.013508 *
## pas3:taskquadruplet -0.20902    0.05287  -3.954 7.69e-05 ***
## pas4:taskquadruplet -0.21500    0.05230  -4.111 3.94e-05 ***
## pas2:tasksingles     0.19536    0.04830   4.045 5.23e-05 ***
## pas3:tasksingles     0.24299    0.05369   4.526 6.02e-06 ***
## pas4:tasksingles     0.56346    0.05101  11.045  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) pas2   pas3   pas4   tskqdr tsksng ps2:tskq ps3:tskq
## pas2        -0.742
## pas3        -0.829  0.613
## pas4        -0.847  0.412  0.703
## taskqudrplt -0.151  0.138  0.080  0.057
## tasksingles -0.138  0.126  0.073  0.052  0.484
## ps2:tskqdrp  0.102 -0.198 -0.054 -0.039 -0.679 -0.328
## ps3:tskqdrp  0.089 -0.082 -0.125 -0.034 -0.592 -0.286  0.402
## ps4:tskqdrp  0.090 -0.083 -0.048 -0.093 -0.598 -0.289  0.406    0.354
## ps2:tsksngl  0.098 -0.188 -0.052 -0.037 -0.342 -0.708  0.490    0.203
## ps3:tsksngl  0.088 -0.080 -0.124 -0.033 -0.308 -0.637  0.209    0.486
## ps4:tsksngl  0.092 -0.085 -0.049 -0.091 -0.324 -0.670  0.220    0.192
##             ps4:tskq ps2:tsks ps3:tsks
## pas2
## pas3
## pas4
## taskqudrplt
## tasksingles
## ps2:tskqdrp
## ps3:tskqdrp
## ps4:tskqdrp
## ps2:tsksngl  0.205
## ps3:tsksngl  0.184    0.451
## ps4:tsksngl  0.507    0.474    0.427
```

```
iv. when you have a converging fit - fit a model with only the main effects of _pas_
and _task_. Compare this with the model that also includes the interaction
```

```
#now making a model of count with no interaction, but pas and task as fixed effects:
m_count2 <- glmer(count~ task + pas +(1+pas|subject), family = poisson, data = data.c
ount, glmerControl(optimizer = "bobyqa"))

summary(m_count2)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: count ~ task + pas + (1 + pas | subject)
##    Data: data.count
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##   3398.5   3459.8  -1683.3   3366.5      324
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.5885 -0.9001 -0.0477  0.8253  6.5100
##
## Random effects:
##  Groups  Name        Variance Std.Dev. Corr
##  subject (Intercept) 0.3325   0.5766
##          pas2        0.3805   0.6169   -0.75
##          pas3        1.1895   1.0906   -0.84  0.63
##          pas4        2.4221   1.5563   -0.86  0.42  0.73
## Number of obs: 340, groups:  subject, 29
##
## Fixed effects:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     4.004593   0.108722  36.833   <2e-16 ***
## taskquadruplet  0.003294   0.018188   0.181   0.8563
## tasksingles     0.004307   0.018177   0.237   0.8127
## pas2           -0.006532   0.116615  -0.056   0.9553
## pas3           -0.509923   0.204449  -2.494   0.0126 *
## pas4           -0.663839   0.291958  -2.274   0.0230 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr tsksng pas2   pas3
## taskqudrplt -0.084
## tasksingles -0.084  0.501
## pas2        -0.742  0.000  0.000
## pas3        -0.832  0.001  0.000  0.623
## pas4        -0.850 -0.002  0.000  0.412  0.723
```

```
#comparison of the two models:

#by AIC:
AIC <- AIC(m_count_new, m_count2)

#residuals for each model
resid1 <- residuals(m_count_new)
resid2 <- residuals(m_count2)

#comparing by standard deviation of residuals
sd_model1<- sqrt((sum(resid1)^2/length(resid1)-2))
sd_model2<- sqrt((sum(resid2)^2/length(resid2)-2))

#comparing by the residual variance:
resid.var1 <- sum(resid1^2)
resid.var2 <- sum(resid2^2)


residuals_com <- tibble("model" =c("With Interaction", "Without Interaction"), "Res V
ar" = c(resid.var1, resid.var2), "Res SD" = c(sd_model1, sd_model2), "AIC" = c(AIC[1,
2], AIC[2,2]))

residuals_com
```

```
## # A tibble: 2 x 4
##   model               `Res Var` `Res SD`   AIC
##   <chr>                   <dbl>    <dbl> <dbl>
## 1 With Interaction         699.    0.847 3148.
## 2 Without Interaction      962.    1.32  3399.
```

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

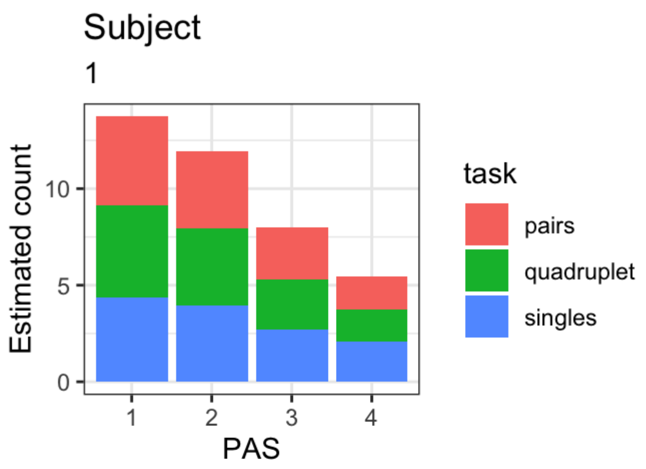
*Comparing the models by AIC, the standard deviation of residuals, and the residual variance, I would choose the first model with interaction (glmer(count ~ pas * task + (1+pas|subject)) since it in both cases has the smaller score. This model does also include the interaction, which could be an argument for the choice of model as well. One could imagine a relationship between the type of cue given and the confidence of the subject in the pas.*

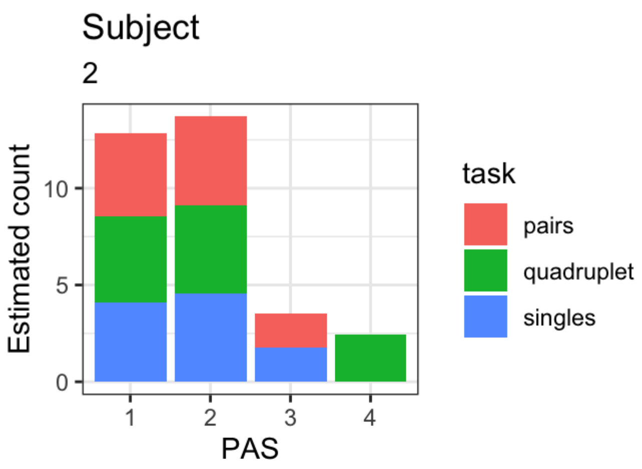
vi. based on your chosen model - write a short report on what this says about the distribution of ratings as dependent on *pas* and *task*

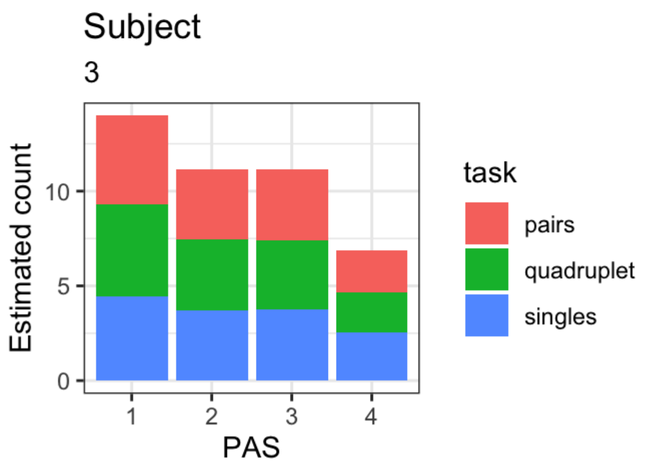
The analysis was conducted to investigate the effect of pas and task on the distribution of ratings (count). To test this, I fitted a general linear mixed effect model, with a link function of the Poisson family. This was done in the assumption that count was distributed as a Poisson distribution. My models has the fixed effects pas and task, and the outcome count. The fixed effects are also included as interactions. Moreover, pas is modelled as a random slope for each subject, *meaning that random intercepts are modelled for each subject Moreover, based on the significant estimates of the model output, ratings does not appear to be equally distributed across task and pas, though pas does not solely predict count.*

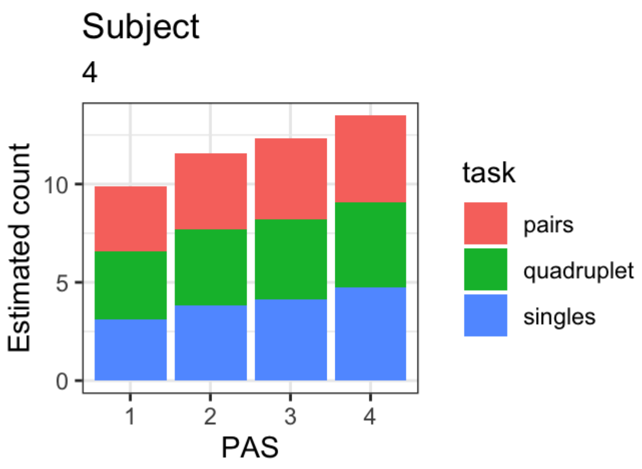
```
vii. include a plot that shows the estimated amount of ratings for four subjects of y
our choosing
```

```
#choosing subject 1, 2, 3 and 4 again :)
subject1 <- data.count %>% filter(subject == "001")
subject2 <- data.count %>% filter(subject == "002")
subject3 <- data.count %>% filter(subject == "003")
subject4 <- data.count %>% filter(subject == "004")


f <- function(i, model){
  sub.pred <- predict(model, newdata = i)
  i$est.count <- sub.pred

  p <- i %>%
    ggplot()+
    geom_bar(aes(x = pas, y = est.count, fill = task), stat = "identity")+
    theme_bw() +
    ggtitle("Subject", as.character(i[1,1])) +
    xlab("PAS")+
    ylab("Estimated count")

}

p1 <- f(subject1, m_count_new)
p2 <- f(subject2, m_count_new)
p3 <- f(subject3, m_count_new)
p4 <- f(subject4, m_count_new)

ggarrange(p1, p2, p3, p4)
```









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

```
model3.1 <- glmer(correct ~ task + (1|subject), family = binomial, data = data)
summary(model3.1)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ task + (1 | subject)
##    Data: data
##
##      AIC      BIC   logLik deviance df.resid
##  19927.2  19958.4  -9959.6  19919.2    18127
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.7426 -1.0976  0.5098  0.6101  0.9111
##
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.1775   0.4214
## Number of obs: 18131, groups:  subject, 29
##
## Fixed effects:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     1.10071    0.08387  13.124  < 2e-16 ***
## taskquadruplet -0.09825    0.04190  -2.345    0.019 *
## tasksingles     0.18542    0.04336   4.276  1.9e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr
## taskqudrplt -0.256
## tasksingles -0.247  0.495
```

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

to test if it does we must take the log of the model values, since the model is logistic

```
logit <- function(x) log(x / (1 - x))
inv.logit <- function(x) exp(x) / (1 + exp(x))

taskpair.log <- inv.logit(1.10071)
tasksingles.log <- inv.logit(1.10071 + 0.18542)
taskquad.log <- inv.logit(1.10071 - 0.09825)

tibble("Prob. of Correct in paired task" = taskpair.log, "Prob. of Correct in Singles
Task" = tasksingles.log, "Prob. of Correct in Quadruplets Task" = taskquad.log)
```

```
## # A tibble: 1 x 3
##   `Prob. of Correct in pai… `Prob. of Correct in Sin… `Prob. of Correct in Quad…
##                       <dbl>                     <dbl>                      <dbl>
## 1                     0.750                     0.783                      0.732
```

There is a probability between 70-80% chance to get a correct answer in all three trials, thus I'd argue that the task does not explain performance very well.

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

```
model3.2 <- glmer(correct ~ task + pas + (1|subject), family = binomial, data = data)
summary(model3.2)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ task + pas + (1 | subject)
##    Data: data
##
##      AIC      BIC   logLik deviance df.resid
##  17424.9  17479.5  -8705.5  17410.9    18124
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -8.4872 -0.6225  0.3240  0.5767  1.6144
##
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.1979   0.4449
## Number of obs: 18131, groups:  subject, 29
##
## Fixed effects:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     0.08530    0.09149   0.932    0.351
## taskquadruplet -0.03055    0.04498  -0.679    0.497
## tasksingles    -0.01059    0.04688  -0.226    0.821
## pas2            0.95477    0.04420  21.599   <2e-16 ***
## pas3            1.97709    0.06221  31.780   <2e-16 ***
## pas4            3.12732    0.08626  36.254   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr tsksng pas2   pas3
## taskqudrplt -0.260
## tasksingles -0.226  0.489
## pas2        -0.213  0.021 -0.040
## pas3        -0.166  0.030 -0.045  0.356
## pas4        -0.123  0.016 -0.079  0.257  0.236
```

Now that pas is added as a main effect along with task, task does not significantly predict correct anymore. Instead pas now do. This could be explained by pas1 now being included in our baseline along with tasksingles, giving that the taskpairs nad taskquadruplet estimates is now only compared with pas 1 .. maybe.

```
iii. now fit a multilevel model that models _correct_ as dependent on _pas_ with a un
ique intercept for each _subject_
```

```
model3.3<- glmer(correct ~ pas + (1|subject), data = data, family = "binomial")
model3.3
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ pas + (1 | subject)
##    Data: data
##       AIC       BIC    logLik  deviance  df.resid
## 17421.387 17460.414 -8705.694 17411.387     18126
## Random effects:
##  Groups  Name        Std.Dev.
##  subject (Intercept) 0.4451
## Number of obs: 18131, groups:  subject, 29
## Fixed Effects:
## (Intercept)         pas2         pas3         pas4
##     0.07044      0.95575      1.97892      3.12940
```

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

```
model3.4 <- glmer(correct ~ pas*task + (1|subject), data = data, family = "binomial")
model3.4
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ pas * task + (1 | subject)
##    Data: data
##       AIC       BIC    logLik  deviance  df.resid
## 17430.974 17532.444 -8702.487 17404.974     18118
## Random effects:
##  Groups  Name        Std.Dev.
##  subject (Intercept) 0.4458
## Number of obs: 18131, groups:  subject, 29
## Fixed Effects:
##         (Intercept)                 pas2                 pas3
##            0.083740             0.963210             1.999968
##                pas4       taskquadruplet          tasksingles
##            3.049586             0.006492            -0.058967
## pas2:taskquadruplet  pas3:taskquadruplet  pas4:taskquadruplet
##           -0.049301            -0.134143            -0.095786
##    pas2:tasksingles     pas3:tasksingles     pas4:tasksingles
##            0.040476             0.079941             0.296579
```

```
v. describe in your words which model is the best in explaining the variance in accur
acy
```

```
#comparison of the  models:
#by AIC:
AIC1 <- AIC(model3.1, model3.2, model3.3, model3.4)

#residuals for each model
resid3.1 <- residuals(model3.1)
resid3.2 <- residuals(model3.2)
resid3.3 <- residuals(model3.3)
resid3.4 <- residuals(model3.4)

#comparing by standard deviation of residuals
sd_model3.1<- sqrt((sum(resid3.1)^2/length(resid3.1)-2))
sd_model3.2<- sqrt((sum(resid3.2)^2/length(resid3.2)-2))
sd_model3.3<- sqrt((sum(resid3.3)^2/length(resid3.3)-2))
sd_model3.4<- sqrt((sum(resid3.4)^2/length(resid3.4)-2))

#comparing by the residual variance:
resid.var3.1 <- sum(resid3.1^2)
resid.var3.2 <- sum(resid3.2^2)
resid.var3.3 <- sum(resid3.3^2)
resid.var3.4 <- sum(resid3.4^2)

library(tidyverse)

residuals_comparison <- tibble("model" =c("Model 3.1", "Model 3.2", "Model 3.3", "Mod
el 3.4"), "Res Var" = c(resid.var3.1, resid.var3.2, resid.var3.3, resid.var3.4), "Res
SD" = c(sd_model3.1, sd_model3.2, sd_model3.3, sd_model3.4), "AIC" = c(AIC1[1,2], AIC
1[2,2], AIC1[3,2], AIC1[4,2]))

residuals_comparison
```

```
## # A tibble: 4 x 4
##   model     `Res Var` `Res SD`    AIC
##   <chr>         <dbl>    <dbl>  <dbl>
## 1 Model 3.1    19804.     19.9 19927.
## 2 Model 3.2    17297.     16.6 17425.
## 3 Model 3.3    17297.     16.6 17421.
## 4 Model 3.4    17291.     16.6 17431.
```

*I would suggest model 3.4: (correct ~ pas***task + (1|subject), family = “binomial”), since this has both the lowest residual variance (17294.14) and standard deviation of residuals (16.74). Thus the model must explain much of the variance it self. Intuitively, this makes rather good sense since one could imagine how subjects experience of the PAS interact with the type of task.*