Assignment 2 (Part 1), Methods 3, 2021, autumn semester

Aleksander Moeslund Wael

01/10/2021

Loading packages

```
pacman::p_load(tidyverse, lmerTest, lme4, gridExtra, dfoptim)
```

Exercise 1

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

```
files <- list.files(path = "experiment_2",
                    pattern = ".csv",
                    full.names = T)

df <- read_csv(files)
```

```
## Rows: 18131 Columns: 17
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): trial.type, task, target.type, obj.resp, subject
## dbl (12): pas, trial, jitter.x, jitter.y, odd.digit, target.contrast, target...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

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

The dataset contains 18131 observations measured on 17 variables. Data from 29 subjects is included. The data originates from a study on measuring subjective experience in a visual task, where participants have to rate a visual stimulus based on the Perceptual Awareness Scale (PAS). Thorough description of the variables are in next task.

1.2.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.

Adding variable “correct” to display if subject was correct

```
# Adding empty variable
df <- df %>%
  mutate(obj.resp.2 = obj.resp)

# Renaming rows in obj.resp.2 to get same units as target.type
df$obj.resp.2 <- replace(df$obj.resp.2, df$obj.resp.2 == "e", "even")
df$obj.resp.2 <- replace(df$obj.resp.2, df$obj.resp.2 == "o", "odd")

# Adding value for correct and incorrect answers
df_correct <- df %>%
  filter(obj.resp.2 == target.type) %>%
  mutate(correct = "1")

# Joining with my df
df <- left_join(df, df_correct)
```

```
## Joining, by = c("trial.type", "pas", "trial", "jitter.x", "jitter.y", "odd.digit", "target.contrast", "target.frames", "c
ue", "task", "target.type", "rt.subj", "rt.obj", "even.digit", "seed", "obj.resp", "subject", "obj.resp.2")
```

```
# Remaining are NAs, so replace with 0
df$correct <- replace(df$correct, is.na(df$correct), "0")
```

1.2.ii. Describe what the following variables in the data frame contain, trial.type, pas, 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 variables 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: Indicates whether subject is doing the staircase task (first experiment) or the follow-up experiment. Should be class character, as it is a category. pas: Indicates subjects response to trial on the Perceptual Awareness Scale (PAS). It is treated as factor. trial: A numbered list for every trial the subject completes, i.e. presses e or o in either of the trial types., per subject. I should think character class for now (might change). target.contrast: The contrast between the background and the digit (target). Between 0-1, treated as numeric. cue: The specific cue pattern, will treat as character. task: Whether cue pattern is 2 (singles), 4 (pairs) or 8 (quadruplets) digits. Will treat as character. target.type: Whether target type is an odd or even number - will treat as character. rt.subj: Reaction time for response to PAS pr. trail - will treat as numeric. rt.obj: Reaction time for responding if target is even or odd - will treat as numeric. obj.resp: Subjects response to target is either even or odd - will treat as character. subject: Participant ID, ordered from 001. Treated as character/factor. correct: Whether subject answered correctly in the trail, 1 for correct and 0 for incorrect. Is logical (binary), NOTE: treated as a factor due to an error when conducting analysis.

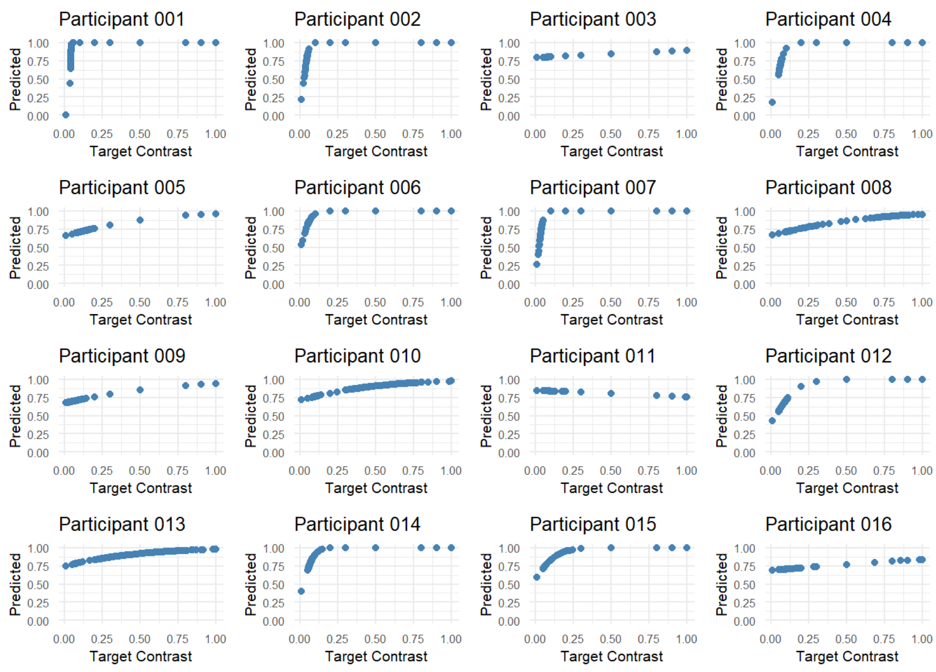
```
# Assigning variables to proper class
df$pas <- as.factor(df$pas)
df$trial <- as.character(df$trial)
df$target.contrast <- as.numeric(df$target.contrast)
df$cue <- as.character(df$cue)
df$rt.subj <- as.numeric(df$rt.subj)
df$rt.obj <- as.numeric(df$rt.obj)
df$target.contrast <- as.numeric(df$target.contrast)
df$correct <- as.factor(df$correct)
df$subject <- as.factor(df$subject)
```

1. 2. 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 enough data to plot the logistic functions?

```
# Making a df
staircase <- df %>%
  filter(df$trial.type == 'staircase')

# No pooling function that returns plot pr. participant
np_function <- function(i){
  dat <- staircase[which(staircase$subject == i),] # subsetting
  model <- glm(correct ~ target.contrast, family = 'binomial', data = dat) # running per participant
  fitted <- model$fitted.values # fitted values
  plot_dat <- data.frame(cbind(fitted, 'target.contrast' = dat$target.contrast)) # append to df with my variables

  plot <- ggplot(plot_dat, aes(x = target.contrast, y = fitted))+
    geom_point(color = 'steelblue') +
    xlab('Target Contrast') +
    ylab('Predicted') +
    ylim(c(0,1))+
    ggtitle(paste0('Participant ', as.character(i))) +
    theme_minimal() +
    theme(plot.title = element_text(size = 10), axis.title=element_text(size = 8), axis.text=element_text(size=6))

  return(plot)
}

# Applying function to subjects and plotting with grid.arrange
subjects <- c("001", "002", "003", "004", "005", "006", "007", "008", "009", "010", "011", "012", "013", "014", "015", "016"
)
plots <- lapply(subjects, FUN=np_function)
```

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

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

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

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

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

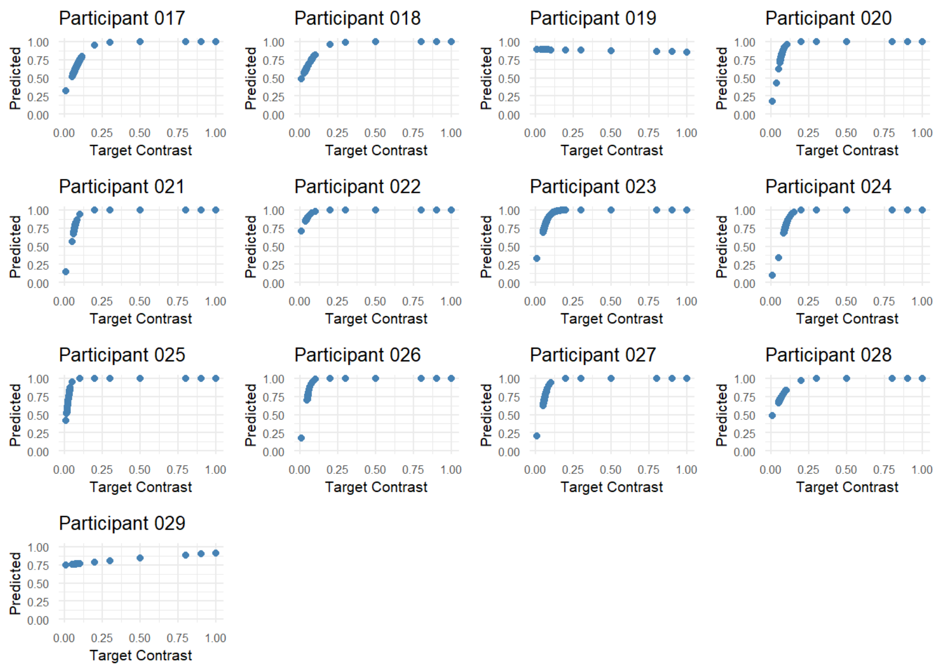
```
do.call(grid.arrange,  plots)
```



```
# With remaining participants
subjects <- c("017", "018", "019", "020", "021", "022", "023", "024", "025", "026", "027", "028", "029")
plots <- lapply(subjects, FUN=np_function)
```

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

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

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

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

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

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

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

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

```
do.call(grid.arrange,  plots)
```



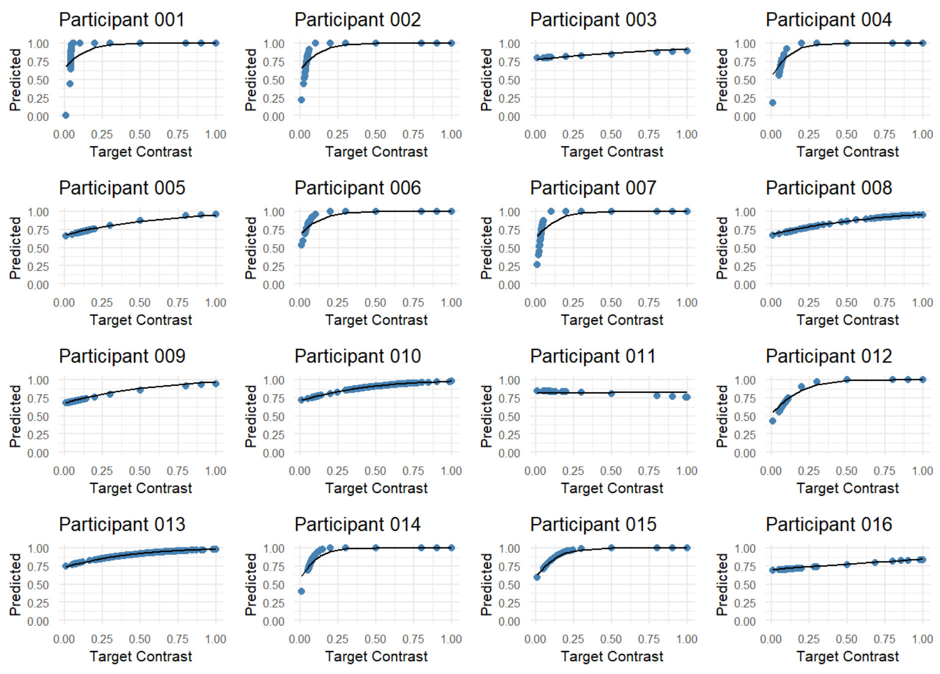
We have ~180 rows pr subject, which is pretty good, but it isn't a meaningful analysis. The no pooling analysis does not inform us about the population, but shows that there is between-subject variance, seen by the difference in sigmoid-shapes for each plot.

1. 2. iv. on top of those plots, add the estimated functions (on the target.contrast range from 0-1) for each subject based on partial pooling model (use glmer from the package lme4) where unique intercepts and slopes for target.contrast are modelled for each subject

```r
# Making a df
staircase <- df %>%
  filter(df$trial.type == 'staircase')

# Model with random intercepts and slopes (partial pooling)
m2 <- glmer(correct ~ target.contrast + (1 + target.contrast | subject), family = 'binomial', data = staircase)

# Function from before, but altered to add estimated functions
pp_function <- function(i){
  dat <- staircase[which(staircase$subject == i),]
  model <- glm(correct ~ target.contrast, family = 'binomial', data = dat)
  fitted <- model$fitted.values
  plot_dat <- data.frame(cbind(fitted, 'target.contrast' = dat$target.contrast))
  fitted2 <- fitted.values(m2) # Adding fitted values for partial pooling model
  plot_dat_2 <- staircase %>% # Subsetting this df also pr subject
    mutate("fitted.values" = fitted2) %>%
    filter(subject == i)

  plot <- ggplot(plot_dat, aes(x = target.contrast, y = fitted))+
    geom_point(color = 'steelblue') +
    geom_line(data = plot_dat_2, aes(x = target.contrast, y = fitted.values)) + # THIS IS THE PARTIAL POOLING MODEL
    xlab('Target Contrast') +
    ylab('Predicted') +
    ylim(c(0,1))+
    ggtitle(paste0('Participant ', as.character(i))) +
    theme_minimal() +
    theme(plot.title = element_text(size = 10), axis.title=element_text(size = 8), axis.text=element_text(size=6))

  return(plot)
}

# Applying function to subjects and plotting with grid.arrange
subjects <- c("001", "002", "003", "004", "005", "006", "007", "008", "009", "010", "011", "012", "013", "014", "015", "016"
)
plots <- lapply(subjects, FUN=pp_function)
```

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

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

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

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

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

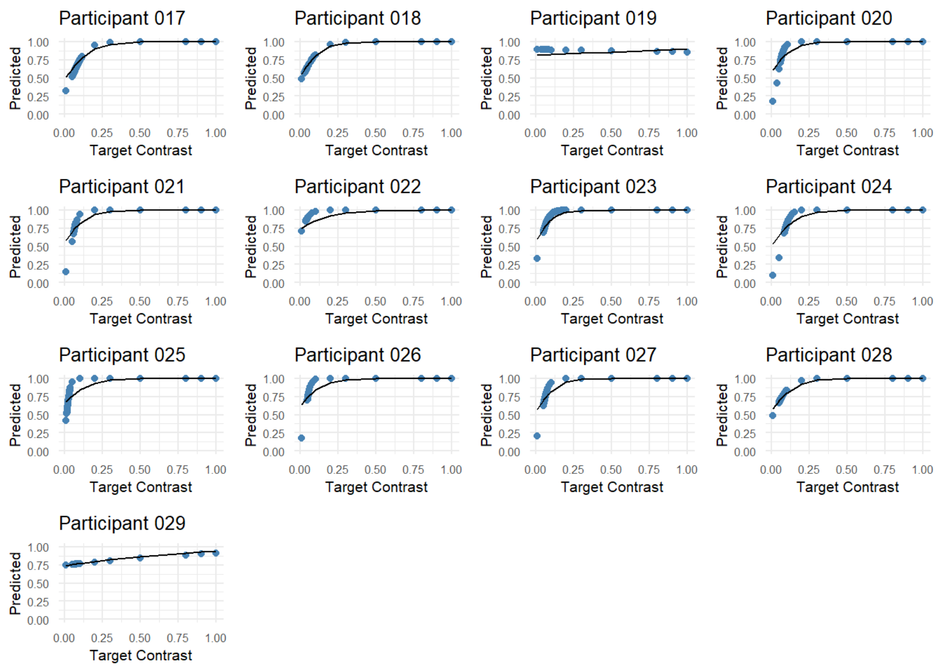
```r
# With remaining participants
do.call(grid.arrange,  plots)
```



```
subjects <- c("017", "018", "019", "020", "021", "022", "023", "024", "025", "026", "027", "028", "029")
plots <- lapply(subjects, FUN=pp_function)
```

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

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

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

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

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

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

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

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

```
do.call(grid.arrange,  plots)
```



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

Partial pooling allows for the model to be generalizable (i.e. "less accurate" fit compared to no pooling), but still accounts for subject differences in baseline (intercept) and performance (slopes). Although it makes little sense to have slopes per subject, as we're only modelling single subject data, but 29 times.

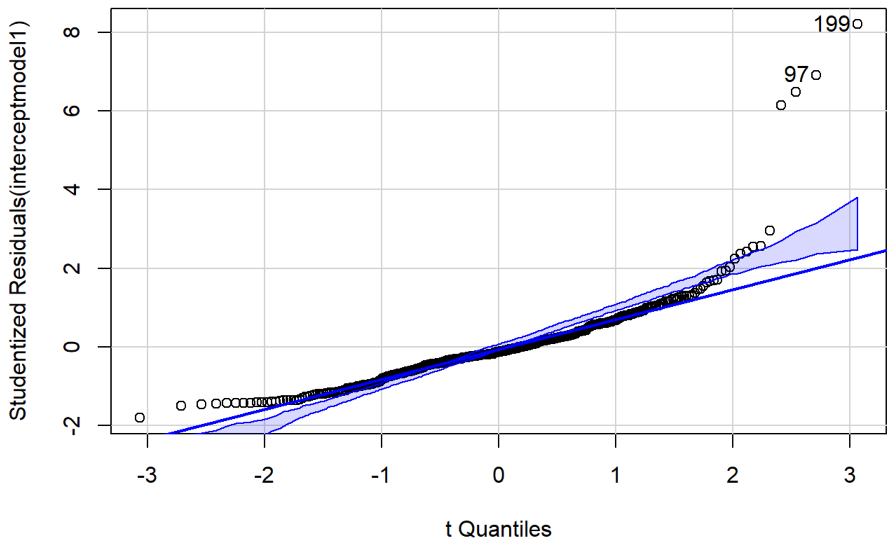
Exercise 2

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

2. 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

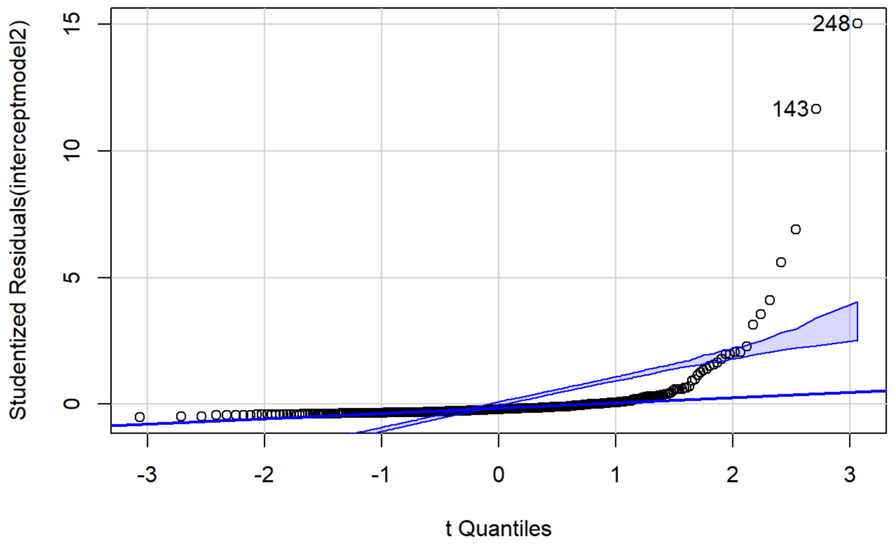
```
# Modelling simple response time
response_time <- lm(rt.obj ~ 1, data = df_experiment)
df_experiment$fitted_rt <- fitted(response_time)
```

```
# Subsetting for each subject
subject1 <- df_experiment %>%
  filter(subject == "001")
subject2 <- df_experiment %>%
  filter(subject == "002")
subject3 <- df_experiment %>%
  filter(subject == "003")
subject4 <- df_experiment %>%
  filter(subject == "004")

# Modelling each subject
interceptmodel1 <- lm(rt.obj ~ 1, data = subject1)
interceptmodel2 <- lm(rt.obj ~ 1, data = subject2)
interceptmodel3 <- lm(rt.obj ~ 1, data = subject3)
interceptmodel4 <- lm(rt.obj ~ 1, data = subject4)

# Plotting
car::qqPlot(interceptmodel1)
```



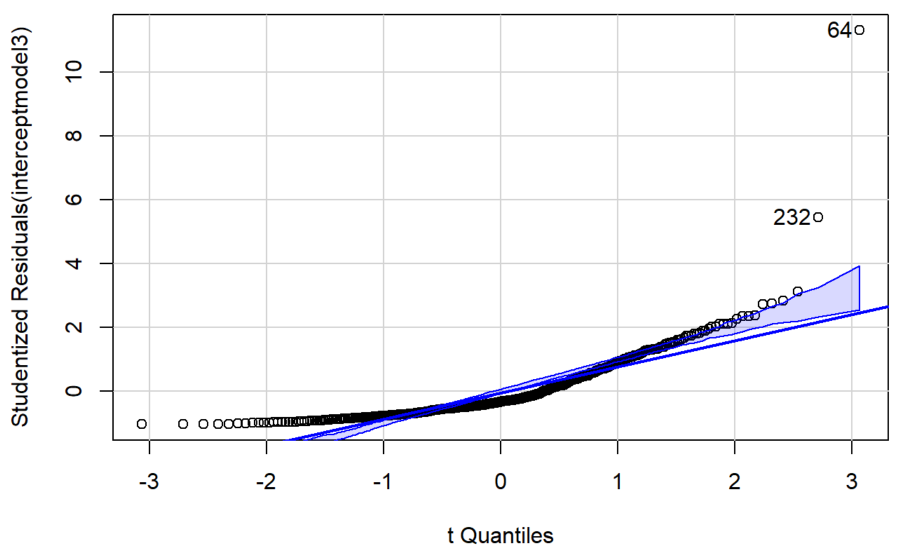
```
## [1]  97 199
```

```
car::qqPlot(interceptmodel2)
```



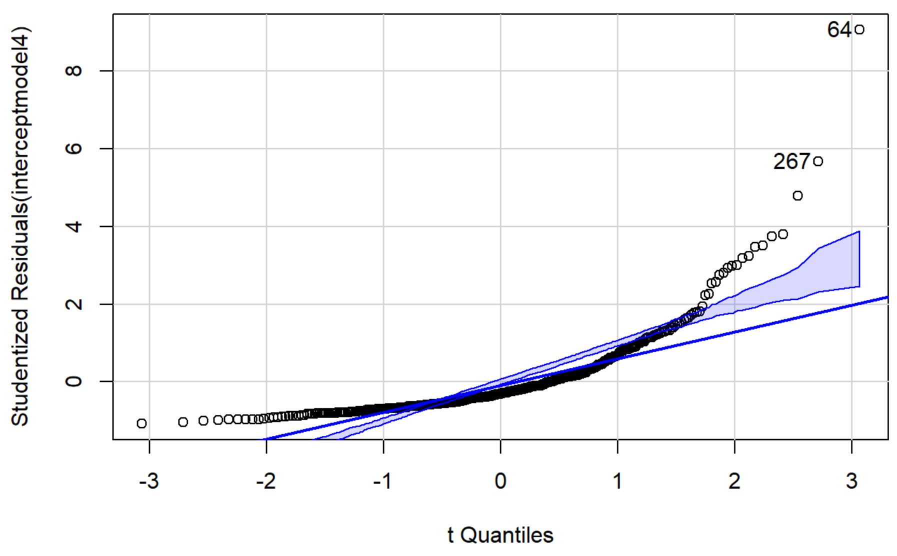
```
## [1] 143 248
```

```
car::qqPlot(interceptmodel3)
```



```
## [1]  64 232
```

```
car::qqPlot(interceptmodel4)
```



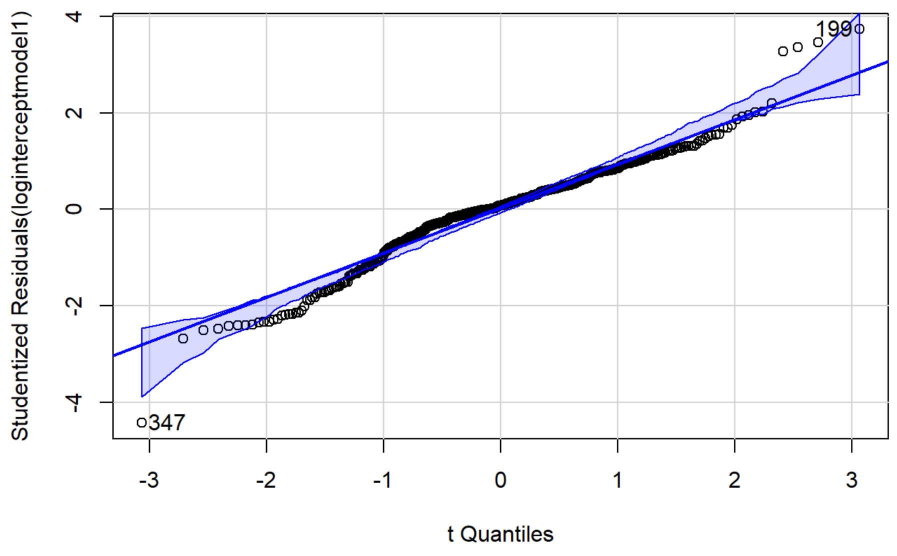
```
## [1]  64 267
```

2. 1. i. comment on these

They are all slightly normal-distributed, but not satisfactory as there is heavy skewness in some of the plots and some heavy outliers in the top end mainly.

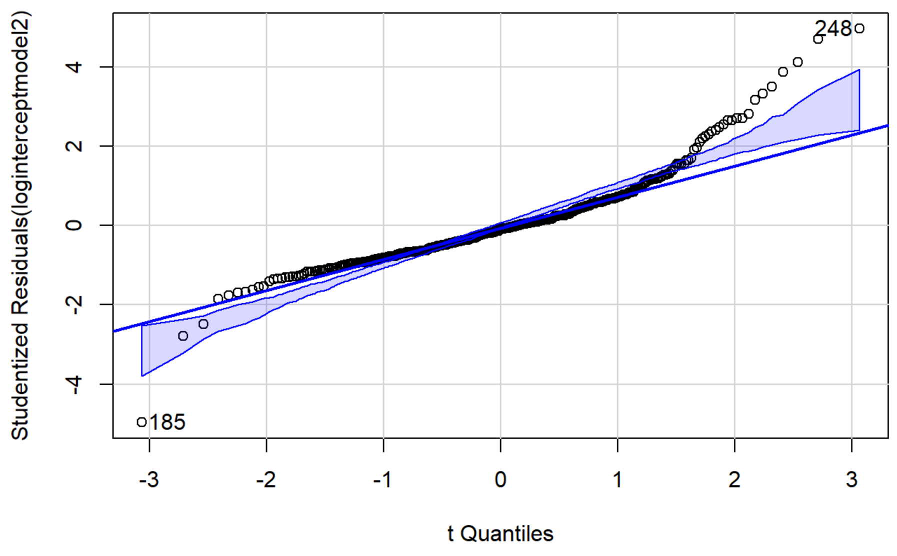
2. 1. ii. does a log-transformation of the response time data improve the Q-Q-plots?

```r
# Log transforming for each subject
logsubject1 <- subject1 %>%
  mutate(log_rt = log(rt.obj))
logsubject2 <- subject2 %>%
  mutate(log_rt = log(rt.obj))
logsubject3 <- subject3 %>%
  mutate(log_rt = log(rt.obj))
logsubject4 <- subject4 %>%
  mutate(log_rt = log(rt.obj))

# Modelling
loginterceptmodel1 <- lm(log_rt ~ 1, data = logsubject1)
loginterceptmodel2 <- lm(log_rt ~ 1, data = logsubject2)
loginterceptmodel3 <- lm(log_rt ~ 1, data = logsubject3)
loginterceptmodel4 <- lm(log_rt ~ 1, data = logsubject4)

# Plotting
car::qqPlot(loginterceptmodel1)
```



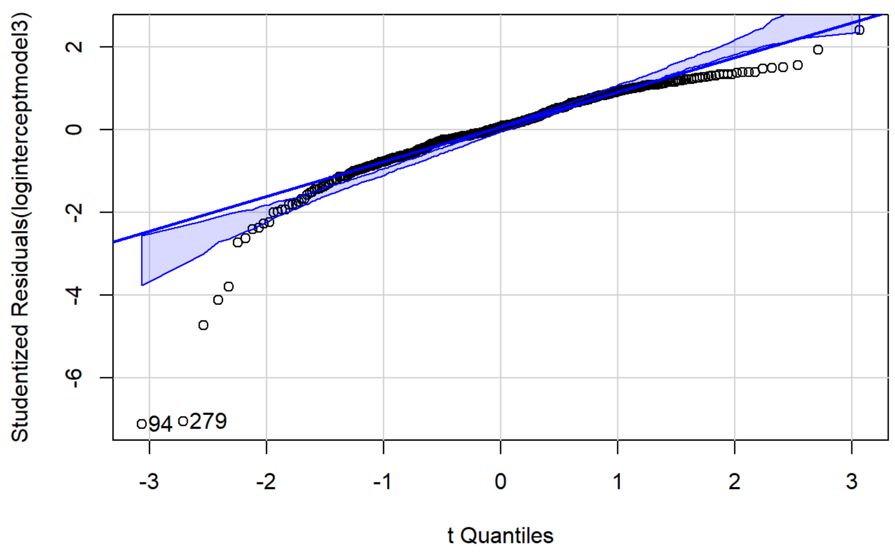
```
## [1] 199 347
```

```r
car::qqPlot(loginterceptmodel2)
```



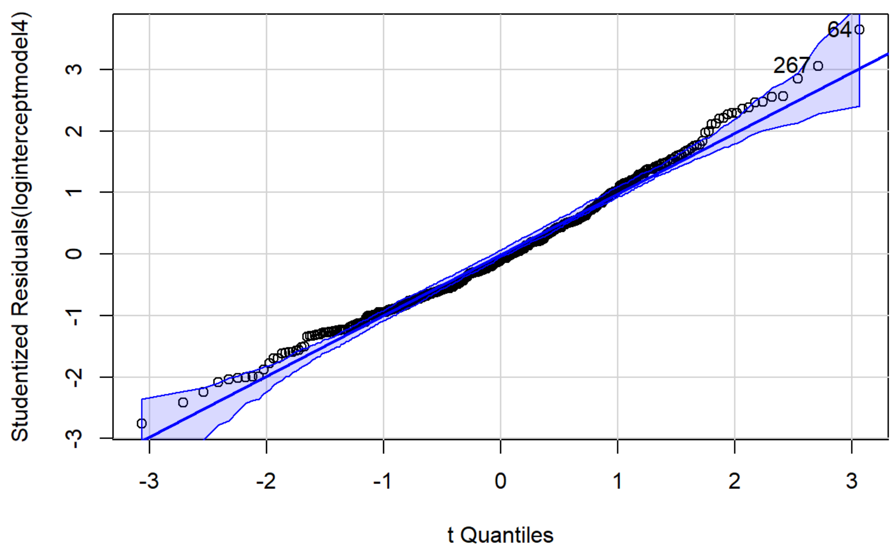
```
## [1] 185 248
```

```r
car::qqPlot(loginterceptmodel3)
```



```
## [1]  94 279
```

```
car::qqPlot(loginterceptmodel4)
```



```
## [1]  64 267
```

I would say the log transformation reduced the tail sizes (outliers), which betters the distribution for subject 1 and 2. Though distribution for subject 3 is now less normal than before transformation.

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

```
rt_partialpooling1 <- lmer(rt.obj ~ task + (1|subject), REML = FALSE, data = df_experiment)
rt_partialpooling2 <- lmer(rt.obj ~ task + (1|subject) + (1|trial), REML = FALSE, data = df_experiment)
```

2. 2. 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)

```
rt_no_ranef <- lm(rt.obj ~ task, data = df_experiment)
anova(rt_partialpooling1, rt_partialpooling2, rt_no_ranef)
```

	npar	AIC	BIC	logLik	deviance	Chisq
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>

	npar <dbl>	AIC <dbl>	BIC <dbl>	logLik <dbl>	deviance <dbl>	Chisq <dbl>
rt_no_ranef	4	62056.06	62085.81	-31024.03	62048.06	NA
rt_partialpooling1	5	61940.21	61977.39	-30965.11	61930.21	117.84801407
rt_partialpooling2	6	61942.17	61986.78	-30965.08	61930.17	0.04584006

3 rows | 1-7 of 9 columns

```
summary(rt_partialpooling1)
```

```
## Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
##   method [lmerModLmerTest]
## Formula: rt.obj ~ task + (1 | subject)
##    Data: df_experiment
##
##      AIC      BIC   logLik deviance df.resid
##  61940.2  61977.4 -30965.1  61930.2    12523
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
##  -0.630  -0.155  -0.072   0.051 101.443
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  subject  (Intercept) 0.1147   0.3386
##  Residual             8.1739   2.8590
## Number of obs: 12528, groups:  subject, 29
##
## Fixed effects:
##                  Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)     1.120e+00  7.689e-02  4.775e+01  14.568  < 2e-16 ***
## taskquadruplet -1.532e-01  6.257e-02  1.250e+04  -2.449  0.01433 *
## tasksingles    -1.915e-01  6.257e-02  1.250e+04  -3.061  0.00221 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr
## taskqudrplt -0.407
## tasksingles -0.407  0.500
```

Optimally, i should like to model the hierarchies i'd expect exist in the data. That would definitively include random intercepts per subject, as I assume reaction time is in some sense dependent on subject. Although this doesn't explain much of the variance, it does reduce the AIC slightly. I would probably also consider doing random intercepts for trial, but seeing as this increases the AIC value, i don't regard it as the right choice, as i only want to add terms that increase the models explanatory power.

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

My model shows that task significantly predicts reaction time for all three tasks ($p < 0.05$). Subjects have highest reaction time with doubles, then quadruplets, then singles.

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

```
rt_partialpooling3 <- lmer(rt.obj ~ task*pas + (1|subject), REML = FALSE, data = df_experiment)
summary(rt_partialpooling3)
```

```
## Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
##   method [lmerModLmerTest]
## Formula: rt.obj ~ task * pas + (1 | subject)
##    Data: df_experiment
##
##      AIC      BIC   logLik deviance df.resid
##  61917.4  62021.5 -30944.7  61889.4    12514
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
##  -0.667  -0.152  -0.064   0.047 101.556
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  subject  (Intercept) 0.09749  0.3122
##  Residual             8.14982  2.8548
## Number of obs: 12528, groups:  subject, 29
##
## Fixed effects:
##                       Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)          1.335e+00  9.786e-02  1.569e+02  13.644  < 2e-16 ***
## taskquadruplet      -3.229e-01  1.068e-01  1.251e+04  -3.023  0.00251 **
## tasksingles         -2.156e-01  1.155e-01  1.252e+04  -1.866  0.06205 .
## pas2                -1.394e-01  1.150e-01  1.228e+04  -1.213  0.22521
## pas3                -3.359e-01  1.314e-01  1.184e+04  -2.557  0.01058 *
## pas4                -5.795e-01  1.341e-01  9.034e+03  -4.321 1.57e-05 ***
## taskquadruplet:pas2  2.273e-01  1.582e-01  1.252e+04   1.437  0.15067
## tasksingles:pas2     1.251e-01  1.661e-01  1.253e+04   0.753  0.45131
## taskquadruplet:pas3  2.624e-01  1.822e-01  1.252e+04   1.440  0.14979
## tasksingles:pas3     9.453e-02  1.852e-01  1.252e+04   0.510  0.60972
## taskquadruplet:pas4  2.583e-01  1.797e-01  1.251e+04   1.437  0.15068
## tasksingles:pas4     7.097e-02  1.746e-01  1.252e+04   0.406  0.68444
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr tsksng pas2   pas3   pas4   tskq:2 tsks:2 tskq:3
## taskqudrplt -0.562
## tasksingles -0.517  0.474
## pas2        -0.553  0.478  0.441
## pas3        -0.499  0.420  0.386  0.425
## pas4        -0.504  0.410  0.373  0.414  0.395
## tskqdrplt:2  0.378 -0.677 -0.319 -0.697 -0.282 -0.276
## tsksngls:p2  0.368 -0.330 -0.697 -0.672 -0.279 -0.270  0.481
## tskqdrplt:3  0.328 -0.587 -0.279 -0.280 -0.681 -0.238  0.397  0.194
## tsksngls:p3  0.327 -0.296 -0.625 -0.276 -0.677 -0.242  0.199  0.437  0.484
## tskqdrplt:4  0.333 -0.595 -0.282 -0.284 -0.249 -0.658  0.402  0.195  0.348
## tsksngls:p4  0.343 -0.314 -0.662 -0.290 -0.255 -0.681  0.211  0.460  0.185
##             tsks:3 tskq:4
## taskqudrplt
## tasksingles
## pas2
## pas3
## pas4
## tskqdrplt:2
## tsksngls:p2
## tskqdrplt:3
## tsksngls:p3
## tskqdrplt:4  0.176
## tsksngls:p4  0.413  0.507
```

2. 3. i. how many types of group intercepts (random effects) can you add without ending up with convergence issues or singular fits?

```
rt_partialpooling4 <- lmer(rt.obj ~ task*pas + (1|subject) + (1|trial), REML = FALSE, data = df_experiment)
rt_partialpooling5 <- lmer(rt.obj ~ task*pas + (1|subject) + (1|trial) + (1|odd.digit), REML = FALSE, data = df_experiment)
rt_partialpooling6 <- lmer(rt.obj ~ task*pas + (1|subject) + (1|trial) + (1|odd.digit) + (1|cue), REML = FALSE, data = df_ex
periment)
rt_partialpooling7 <- lmer(rt.obj ~ task*pas + (1|subject) + (1|trial) + (1|odd.digit) + (1|cue) + (1|pas), REML = FALSE, da
ta = df_experiment)
```

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

Only the last one failed to converge. It is hard to see in the code chunk, but the model which didn’t converge was the following: rt_partialpooling7 <- lmer(rt.obj ~ task*pas + (1|subject) + (1|trial) + (1|odd.digit) + (1|cue) + (1|pas), REML = FALSE, data = df_experiment)

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

```
print(VarCorr(rt_partialpooling7), comp='Variance')
```

```
##  Groups    Name        Variance
##  trial     (Intercept) 0.00153000
##  cue       (Intercept) 0.08946868
##  subject   (Intercept) 0.09769533
##  pas       (Intercept) 0.00000000
##  odd.digit (Intercept) 0.00016599
##  Residual              8.11287188
```

The fit is singular because random intercept for pas explains no variance, i.e. is estimated at 0, and can therefore not be calculated. This makes sense since i already added pas in the model previously.

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

If you add terms which explain close to zero variance, i.e. too low eigenvalue (below some tolerance level in the function, i assume), the model cannot be fit and fails to converge.

Exercise 3

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

```
# Making df
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.
```

3. 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

```
pasmodel <- glmer(count ~ pas*task + (pas|subject), data = data.count, family = poisson, control = glmerControl(optimizer="b
obyqa"))
summary(pasmodel)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: count ~ pas * task + (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  36.770  < 2e-16 ***
## pas2                 -0.02378    0.11963  -0.199 0.842456    
## pas3                 -0.51365    0.20717  -2.479 0.013164 *  
## pas4                 -0.77292    0.29075  -2.658 0.007851 ** 
## 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
```

3. 2. i. which family should be used?

Poisson, because it's good for modelling frequency, which we are doing with counting the PAS-scores.

3. 2. ii. why is a slope for *pas* not really being modelled?

Because pas isn't continuous, we can't model a proper slope, so this is a "pseudo"-slope.

3. 2. iii. if you get a convergence error, try another algorithm (the default is the *Nelder_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`)

I did get the error, added the optimizer, now it converges.

3. 2. 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

```
pasmodel2 <- glmer(count ~ pas + task + (pas|subject), data = data.count, family = poisson, control = glmerControl(optimizer
="bobyqa"))
anova(pasmodel, pasmodel2)
```

	npar <dbl>	AIC <dbl>	BIC <dbl>	logLik <dbl>	deviance <dbl>	Chisq <dbl>	Df <dbl>
pasmodel2	16	3398.549	3459.812	-1683.274	3366.549	NA	NA
pasmodel	22	3148.441	3232.678	-1552.220	3104.441	262.1076	6

2 rows | 1-8 of 9 columns

The interaction model performs better with all interaction effects being significant (p < 0.05), and with a lower AIC value (p < 0.05).

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

```
tibble("SSR interaction model" = sum(residuals(pasmodel)^2), "SSR no interaction" = sum(residuals(pasmodel2)^2))
```

SSR interaction model <dbl>	SSR no interaction <dbl>
699.4866	962.4628

1 row

I would choose the model with the interaction effect included. The model performs better as shown, and the interaction effects are significant, which is an important feature of the data. The SSR score is also lower for this model.

3. 2. 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*

Building the model

The model i've chosen is count ~ pas * task + (pas | subject). Results from this model indicate that the frequency of observations for a certain combination of PAS-score, task and subject (the count variable) is significantly predicted by PAS-score and task and the interaction between the two (p < 0 .05). The model includes random slopes for PAS-score per subject (intercept).

Results

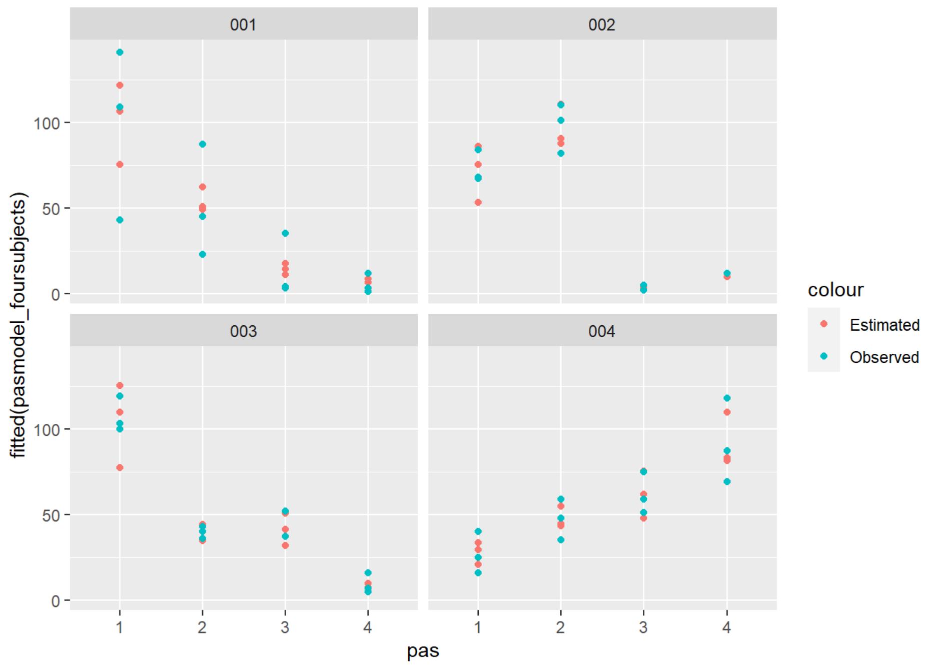
The model suggests that count decreases as PAS-score increases, and count is largest in task "quadruplets", then "doubles" then "singles". Though it is more useful to look at the interaction effects here - when PAS-score increases, the task "quadruplets" decreases compared to "doubles", and "singles" increases compared to "doubles". This effectively means that subjects are actually more "perceptually aware" doing the "singles" task, even though count of PAS-score generally decreases (as per pas main effect) and count of "singles" task generally decreases (as per task main effect).

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

```
# FITTING THE INTERACTION MODEL
# Subsetting
pas_foursubjects <- data.count %>%
  filter(subject == "001"|subject == "002"|subject == "003"|subject == "004")

# Modelling
pasmodel_foursubjects <- glmer(count ~ pas*task + (pas|subject), data = pas_foursubjects, family = poisson)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00304025 (tol = 0.002, component 1)
```

```
# Plotting
pas_foursubjects %>%
  ggplot() +
  geom_point(aes(x = pas, y = fitted(pasmodel_foursubjects), color = "Estimated")) +
  geom_point(aes(x = pas, y = count, color = "Observed")) +
  facet_wrap(~subject)
```



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

```
df_end_me <- glmer(correct ~ task + (1 | subject), data = df, family = "binomial")
```

3. 3. i. does task explain performance?

```
summary(df_end_me)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ task + (1 | subject)
##    Data: df
##
##      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    0.08386  13.125  < 2e-16 ***
## taskquadruplet -0.09825    0.04190  -2.345    0.019 *
## tasksingles      0.18542    0.04337   4.276 1.91e-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
```

Task significantly predicts correctness for all task levels (all p < 0.05).

3. 3. ii. add pas as a main effect on top of task - what are the consequences of that?

```
df_end_me_more <- glmer(correct ~ task + pas + (1 | subject), data = df, family = "binomial")
summary(df_end_me_more)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ task + pas + (1 | subject)
##    Data: df
## 
##      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    0.09143   0.933    0.351    
## taskquadruplet -0.03055    0.04497  -0.679    0.497    
## tasksingles    -0.01059    0.04687  -0.226    0.821    
## pas2            0.95477    0.04419  21.604   <2e-16 ***
## pas3            1.97709    0.06219  31.793   <2e-16 ***
## pas4            3.12732    0.08626  36.255   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Correlation of Fixed Effects:
##             (Intr) tskqdr tsksng pas2   pas3  
## taskqudrplt -0.259                            
## tasksingles -0.225  0.489                     
## pas2        -0.212  0.021 -0.040              
## pas3        -0.165  0.030 -0.045  0.355       
## pas4        -0.123  0.016 -0.080  0.257  0.236
```

Since task is no longer significant, it seems that pas explains more of the variance, i.e. a better predictor.

3. 3. iii. now fit a multilevel model that models correct as dependent on pas with a unique intercept for each subject

```
df_end_me_more_now <- glmer(correct ~ pas + (1 | subject), data = df, family = "binomial")
summary(df_end_me_more_now)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ pas + (1 | subject)
##    Data: df
## 
##      AIC      BIC   logLik deviance df.resid 
##  17421.4  17460.4  -8705.7  17411.4    18126 
## 
## Scaled residuals: 
##     Min      1Q  Median      3Q     Max 
## -8.5665 -0.6243  0.3244  0.5754  1.6017 
## 
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.1981   0.4451  
## Number of obs: 18131, groups:  subject, 29
## 
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept)  0.07044    0.08773   0.803    0.422    
## pas2         0.95575    0.04410  21.671   <2e-16 ***
## pas3         1.97892    0.06201  31.914   <2e-16 ***
## pas4         3.12940    0.08579  36.476   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Correlation of Fixed Effects:
##      (Intr) pas2   pas3  
## pas2 -0.223              
## pas3 -0.173  0.352       
## pas4 -0.136  0.253  0.231
```

3. 3. iv. finally, fit a model that models the interaction between task and pas and their main effects

```
df_end_me_more_now_pls <- glm(correct ~ task * pas, data = df, family = "binomial")
summary(df_end_me_more_now_pls)
```

```
##
## Call:
## glm(formula = correct ~ task * pas, family = "binomial", data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5349  -1.2539   0.5089   0.7922   1.1028
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           0.210096   0.045777   4.590 4.44e-06 ***
## taskquadruplet        0.002246   0.062960   0.036    0.972
## tasksingles          -0.032082   0.068755  -0.467    0.641
## pas2                  0.787873   0.070763  11.134  < 2e-16 ***
## pas3                  1.723013   0.099679  17.286  < 2e-16 ***
## pas4                  2.764730   0.142179  19.445  < 2e-16 ***
## taskquadruplet:pas2 -0.049612   0.098605  -0.503    0.615
## tasksingles:pas2     0.073425   0.103517   0.709    0.478
## taskquadruplet:pas3 -0.108583   0.140751  -0.771    0.440
## tasksingles:pas3     0.077543   0.143470   0.540    0.589
## taskquadruplet:pas4 -0.125402   0.199278  -0.629    0.529
## tasksingles:pas4     0.229071   0.195933   1.169    0.242
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 20418  on 18130  degrees of freedom
## Residual deviance: 17853  on 18119  degrees of freedom
## AIC: 17877
##
## Number of Fisher Scoring iterations: 5
```

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

```
anova(df_end_me, df_end_me_more, df_end_me_more_now, df_end_me_more_now_pls)
```

	npar <dbl>	AIC <dbl>	BIC <dbl>	logLik <dbl>	deviance <dbl>
df_end_me	4	19927.21	19958.43	-9959.605	19919.21
df_end_me_more_now	5	17421.39	17460.41	-8705.694	17411.39
df_end_me_more	7	17424.91	17479.55	-8705.456	17410.91
df_end_me_more_now_pls	12	17877.34	17971.00	-8926.668	17853.34

4 rows | 1-6 of 9 columns

The model which predicts correct by pas with intercepts per subject has the lowest AIC value and a significant chi-square value. This model strikes the balance between complexity and explanatory power, that means it explains the accuracy whilst being simple compared to the other models. Adding predictors without increasing the amount of variance explained doesn’t increase the models performance, but does increase complexity, which isn’t an inherently good thing.