practical_exercise_3, 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. Creating a data frame with all subject data

```
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. Describing the data

The dataset contains 18131 observations described by 17 variables. Data from 29 subjects is included. Thorough description in next task.

1. 2. i.

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"
```

```
# 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). Takes a value between 1-4, and will therefore be treated as numeric. 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.

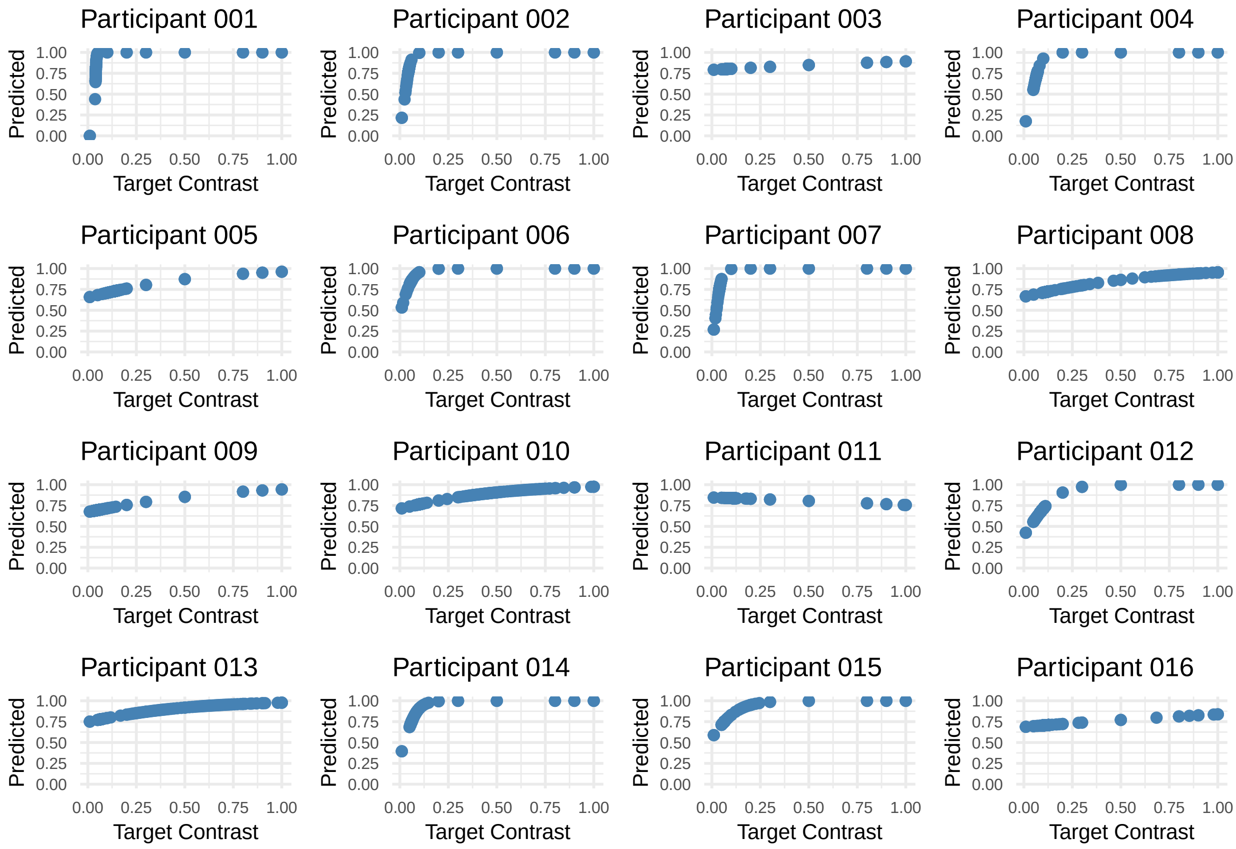
```
# Assigning variables to proper class
df$pas <- as.numeric(df$pas)
df$trial <- as.character(df$trial)
df$target.contrast <- as.numeric(df$target.contrast)
df$cue <- as.character(df$cue)
df$rt.subj <- as.numeric(df$rt.subj)
df$rt.obj <- as.numeric(df$rt.obj)
df$target.contrast <- as.numeric(df$target.contrast)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

# Making a function to run a model for each participant
nopoolfun <- function(i){
  dat <- staircase[which(staircase$subject == i),] # subsetting the data so it only includes one partic
  model <- glm(correct ~ target.contrast, family = 'binomial', data = dat) # running a model on the dat
  fitted <- model$fitted.values # extracting the fitted values
  plot_dat <- data.frame(cbind(fitted, 'target.contrast' = dat$target.contrast)) # creating a data fram

  plot <- ggplot(plot_dat, aes(x = target.contrast, y = fitted))+ # plotting
    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_te

  return(plot)
}

# Running the function for every participant (doing it twice so the plots are nicer to look at)
subjects <- c("001", "002", "003", "004", "005", "006", "007", "008", "009", "010", "011", "012", "013"
plots <- lapply(subjects, FUN=nopoolfun)
```

```
## 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
```

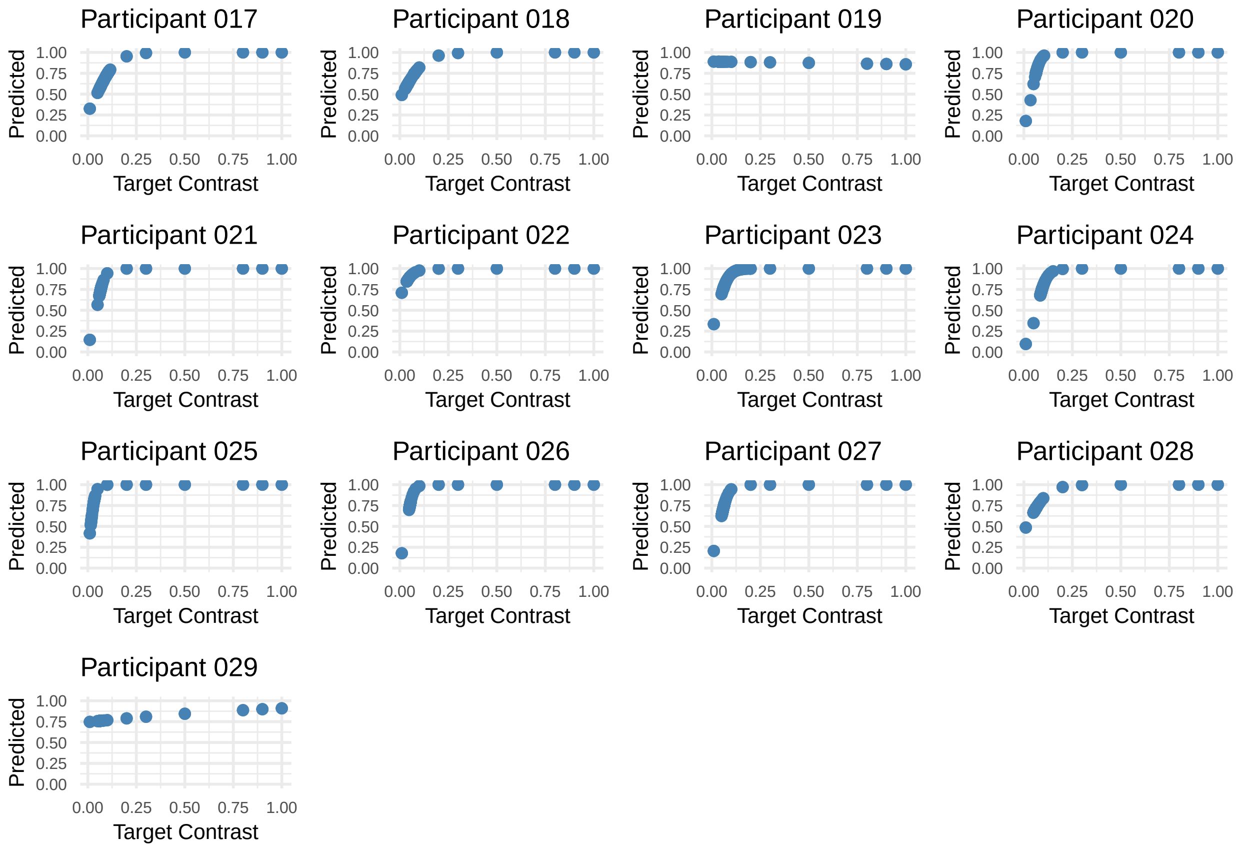
```
do.call(grid.arrange,  plots)
```



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

```
## 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)
```



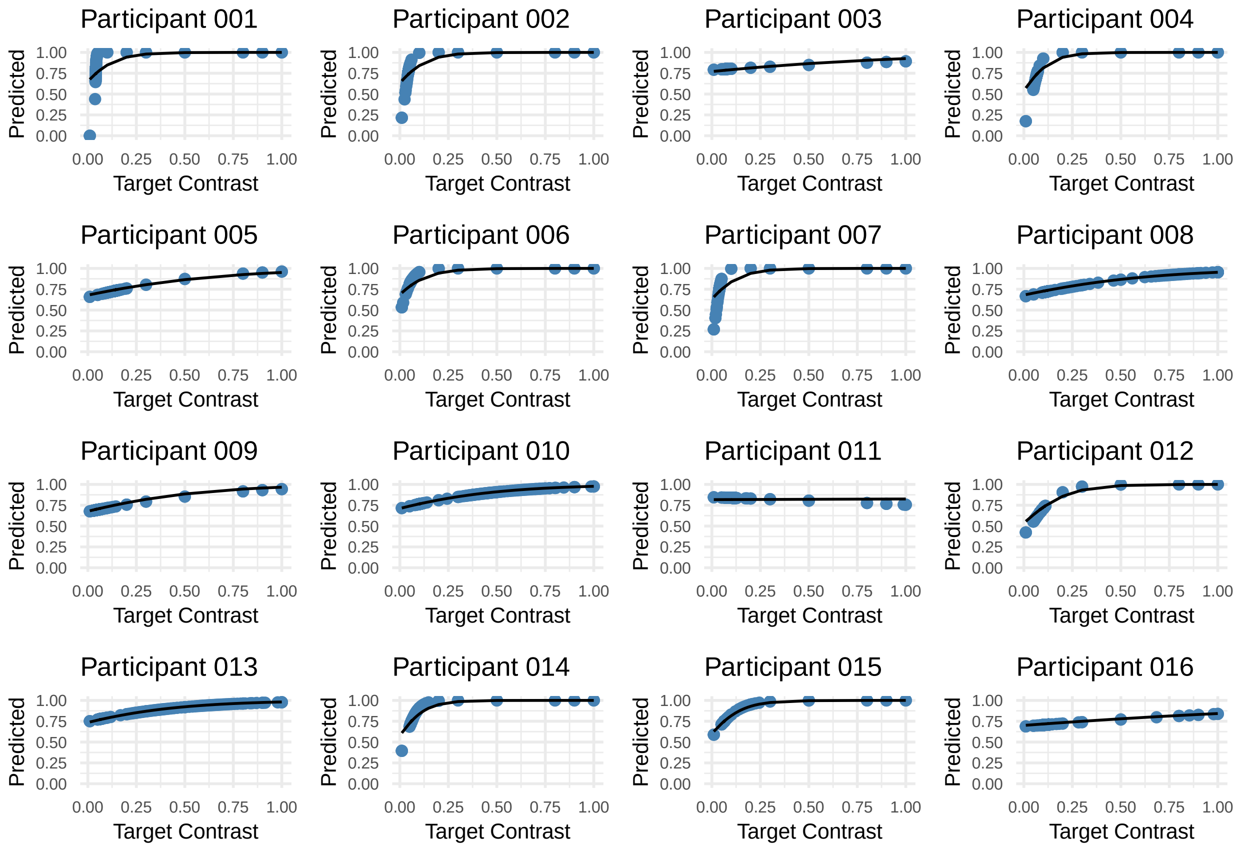
I should think we have enough data, yes, although no pooling analysis does not inform us about the population.

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

```
# 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 = sta

# Making a function to run a model for each participant
partialpoolfun <- function(i){
  dat <- staircase[which(staircase$subject == i),] # subsetting the data so it only includes one partic
  model <- glm(correct ~ target.contrast, family = 'binomial', data = dat) # running a model on the dat
  fitted <- model$fitted.values # extracting the fitted values
  plot_dat <- data.frame(cbind(fitted, 'target.contrast' = dat$target.contrast)) # creating a data fram
  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))+ # plotting
    geom_point(color = 'steelblue') +
    geom_line(data = plot_dat_2, aes(x = target.contrast, y = fitted.values)) + # THIS IS THE PARTIAL PO
    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_tex

  return(plot)
}

# Running the function for every participant
subjects <- c("001", "002", "003", "004", "005", "006", "007", "008", "009", "010", "011", "012", "013"
plots <- lapply(subjects, FUN=partialpoolfun)
```

```
## 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
```

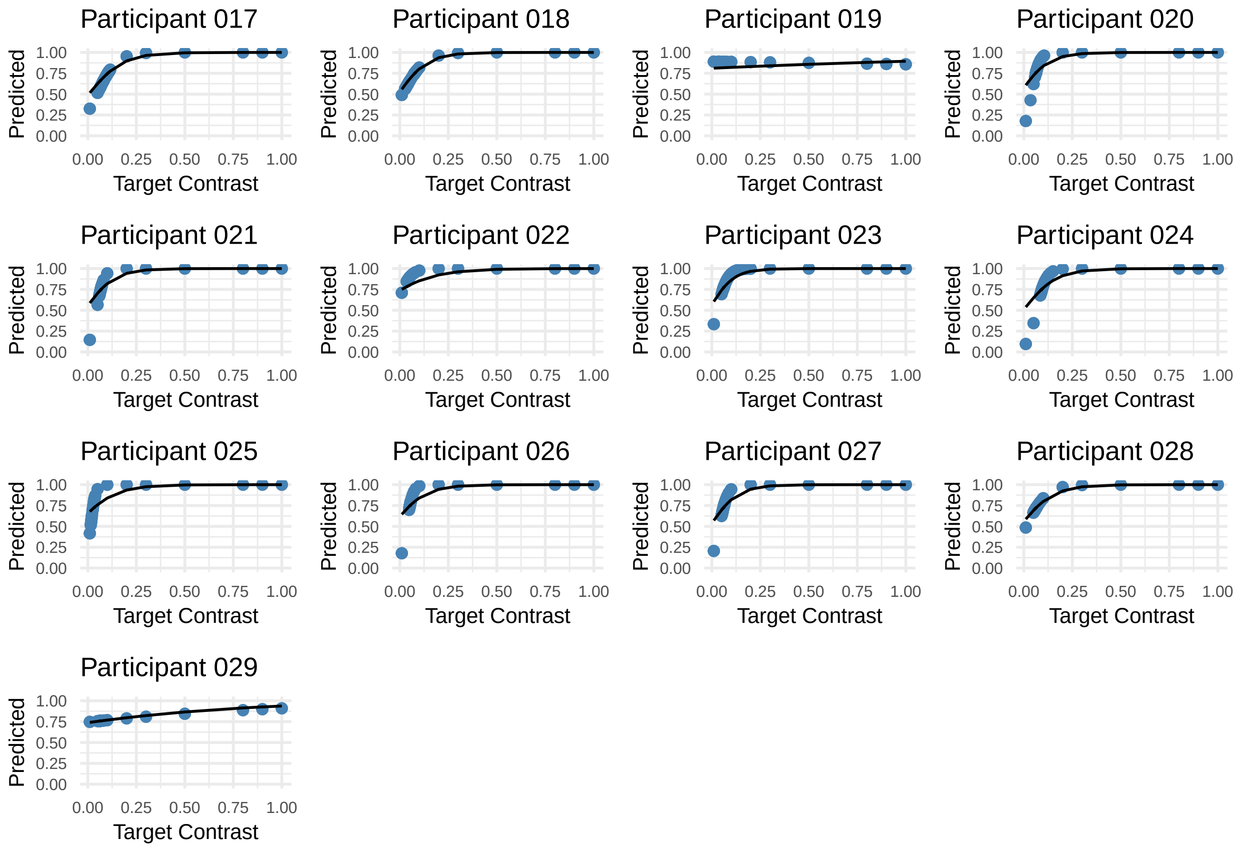
```
do.call(grid.arrange,  plots)
```



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

```
## 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).

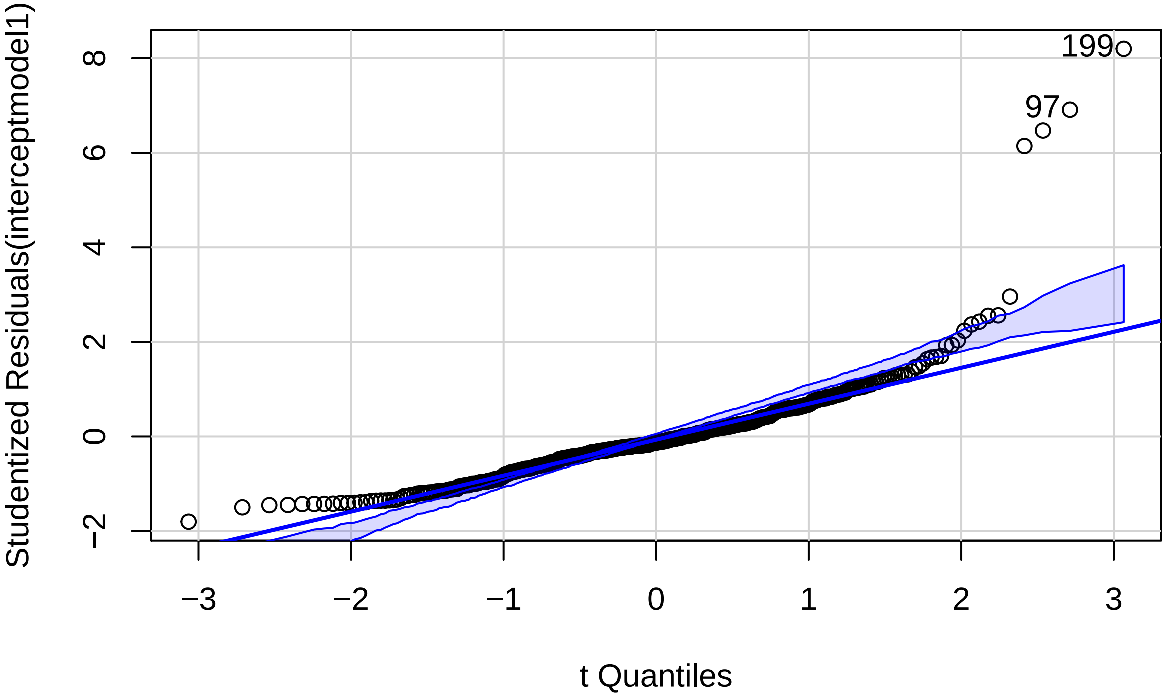
Exercise 2

```r
# 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

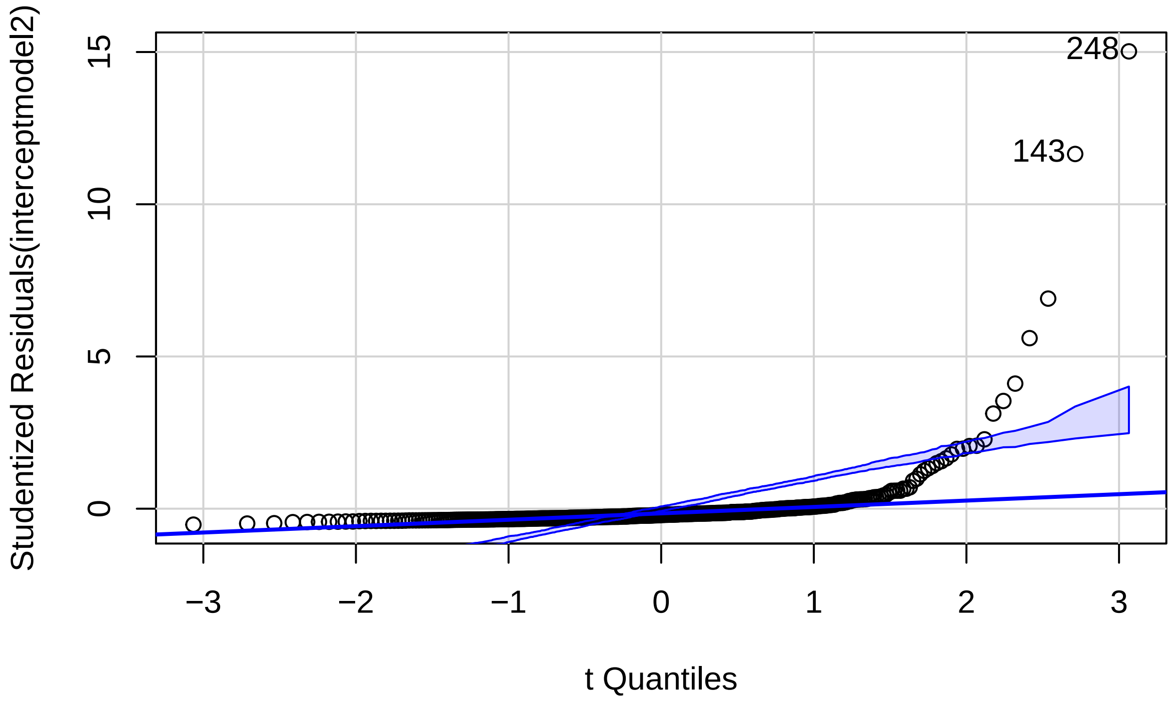
```r
# Modelling simple response time
response_time <- lm(rt.obj ~ 1, data = df_experiment)
df_experiment$fitted_rt <- fitted(response_time)
```

```
# 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)
```



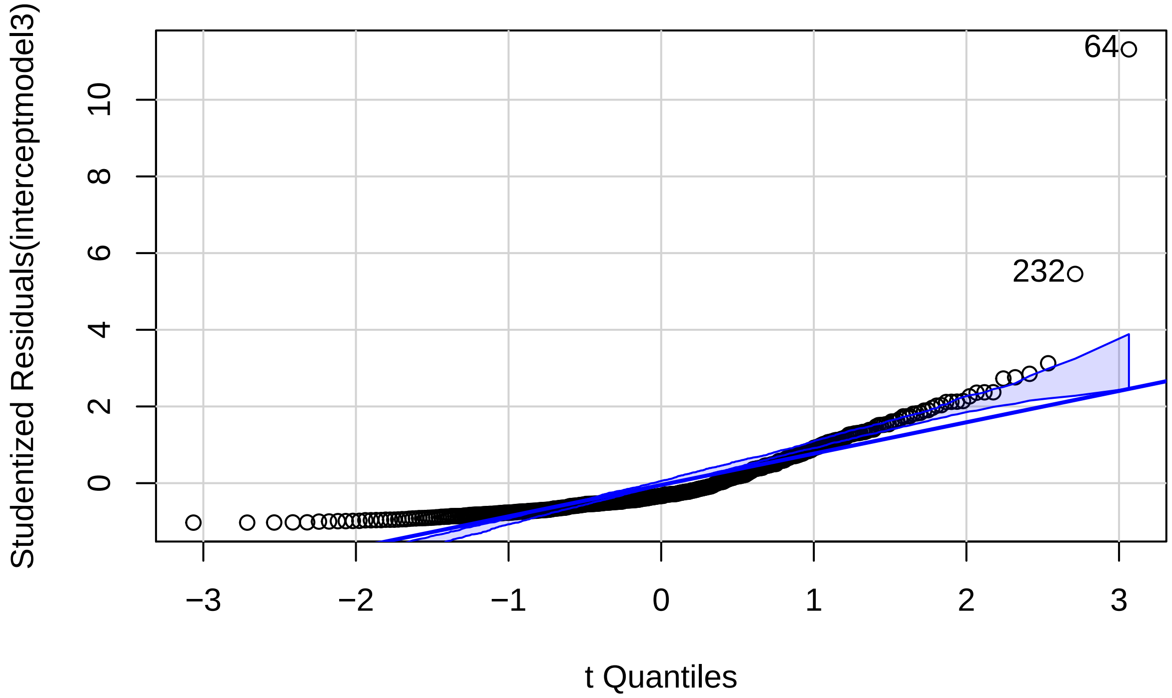
```
## [1]  97 199
```

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



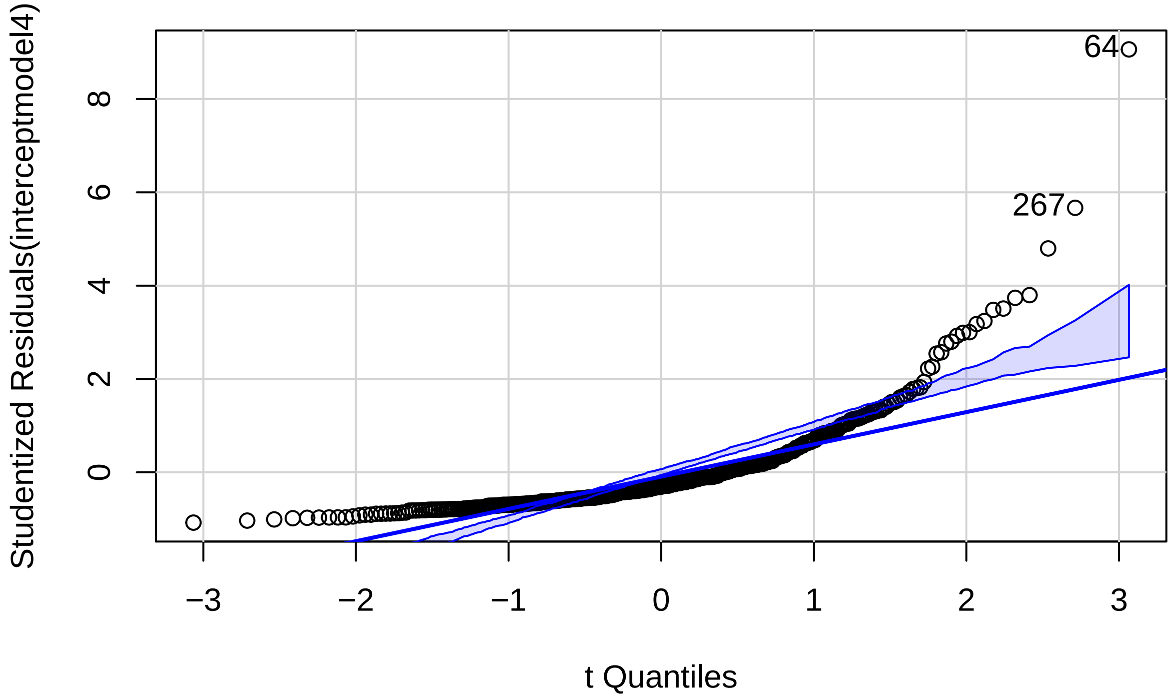
```
## [1] 143 248
```

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



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

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



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

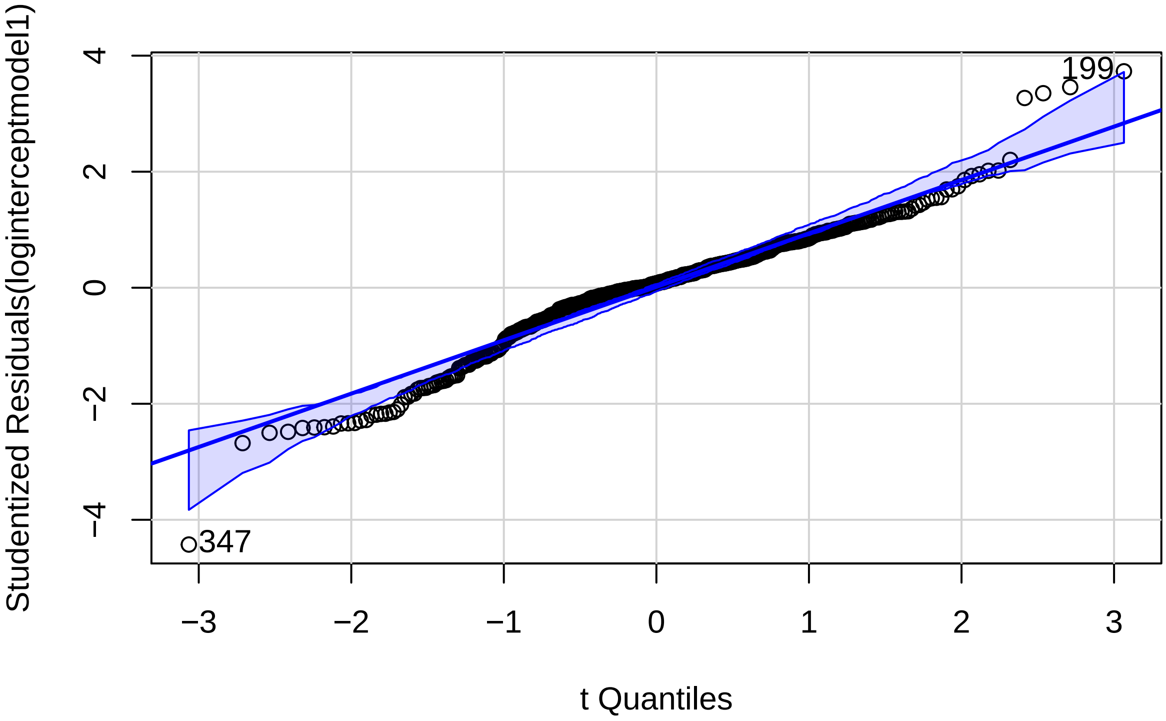
2. 1. i. comment on these

They are all somewhat normally 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?

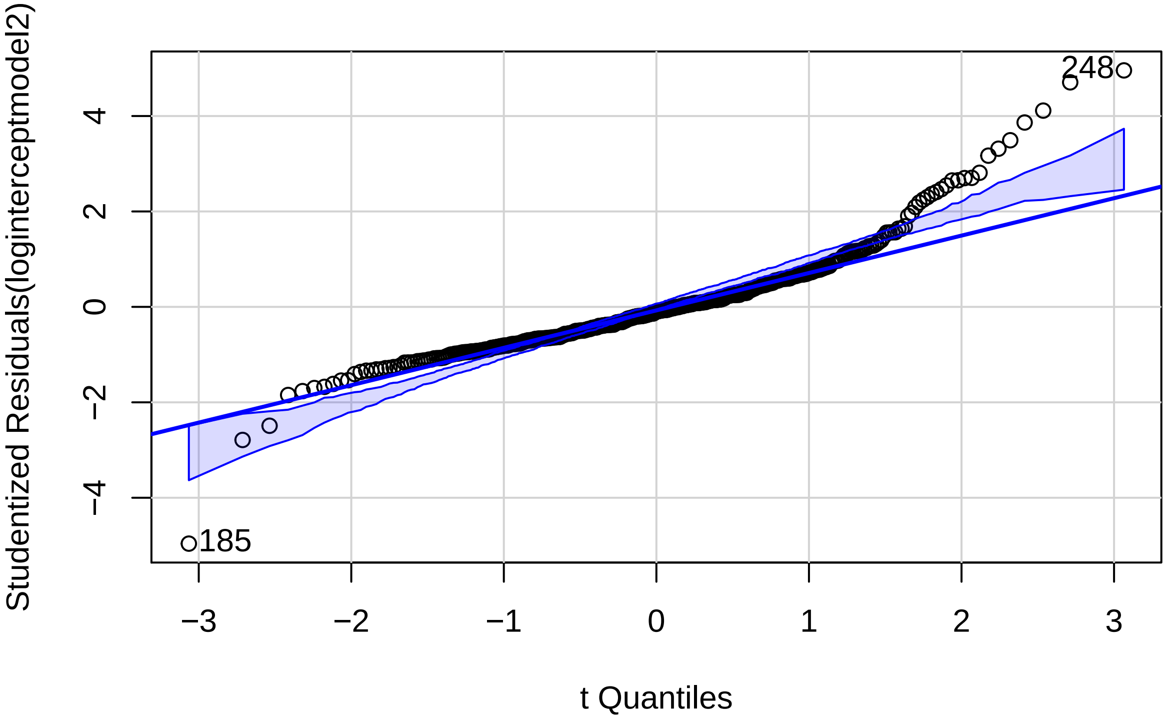
```
# Log transforming for each subject
logsubject1 <- subject1 %>%
  mutate(log_rt = log(rt.obj))
logsubject2 <- subject2 %>%
  mutate(log_rt = log(rt.obj))
logsubject3 <- subject3 %>%
  mutate(log_rt = log(rt.obj))
logsubject4 <- subject4 %>%
  mutate(log_rt = log(rt.obj))

# 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)
```



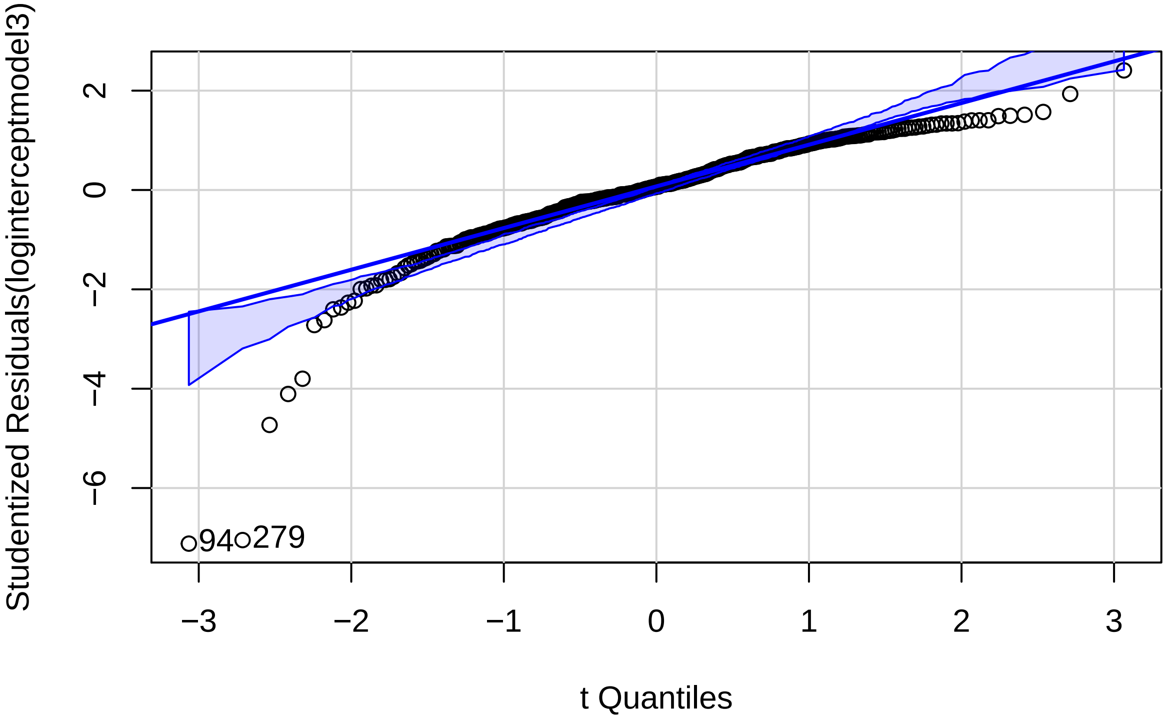
```
## [1] 199 347
```

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



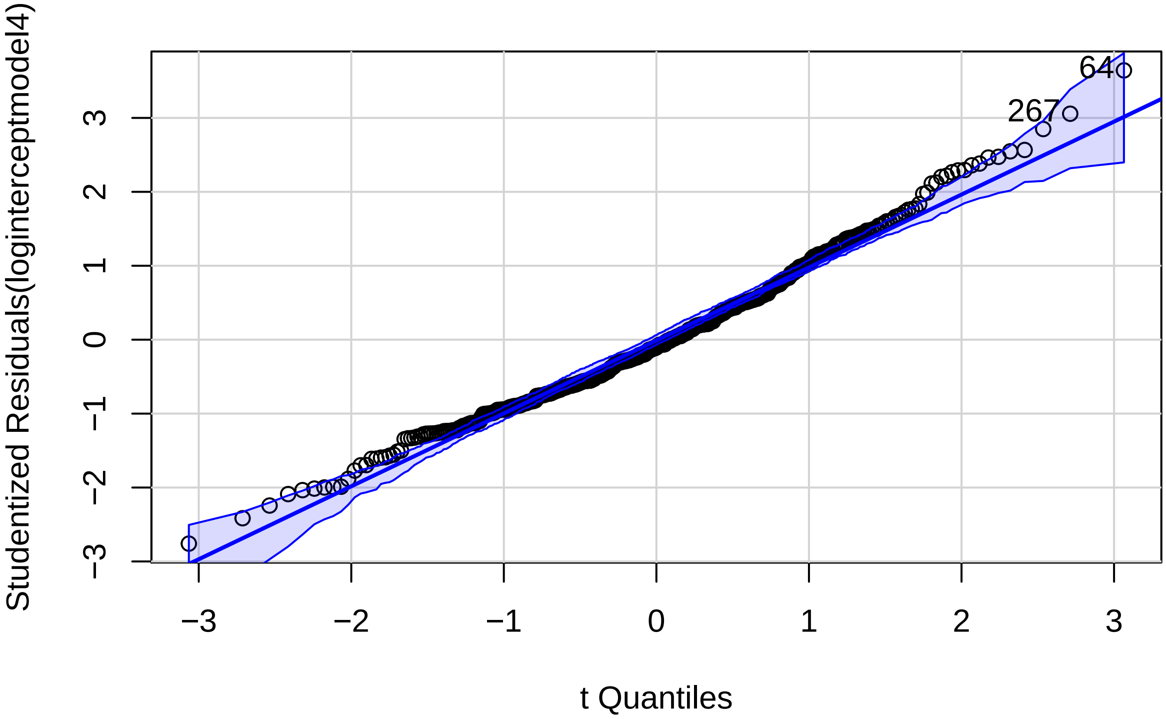
```
## [1] 185 248
```

```
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 optimal.

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)
```

```
## Data: df_experiment
## Models:
## rt_no_ranef: rt.obj ~ task
## rt_partialpooling1: rt.obj ~ task + (1 | subject)
## rt_partialpooling2: rt.obj ~ task + (1 | subject) + (1 | trial)
##                    npar   AIC   BIC logLik deviance    Chisq Df Pr(>Chisq)
## rt_no_ranef           4 62056 62086 -31024    62048
## rt_partialpooling1    5 61940 61977 -30965    61930 117.8480  1     <2e-16 ***
## rt_partialpooling2    6 61942 61987 -30965    61930   0.0458  1     0.8305
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
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 my 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
##  61911.5  61970.9 -30947.7  61895.5    12520
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
##  -0.668  -0.152  -0.064   0.048 101.530
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  subject  (Intercept) 0.09744  0.3122
##  Residual             8.15379  2.8555
## Number of obs: 12528, groups:  subject, 29
##
## Fixed effects:
##                       Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)          1.541e+00  1.197e-01  3.160e+02  12.872  < 2e-16 ***
## taskquadruplet      -3.639e-01  1.397e-01  1.251e+04  -2.605  0.00919 **
## tasksingles         -1.790e-01  1.448e-01  1.252e+04  -1.236  0.21648
## pas                 -1.876e-01  4.234e-02  8.855e+03  -4.431 9.51e-06 ***
## taskquadruplet:pas   9.140e-02  5.646e-02  1.251e+04   1.619  0.10547
## tasksingles:pas      1.312e-02  5.529e-02  1.252e+04   0.237  0.81248
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr tsksng pas    tskqd:
## taskqudrplt -0.592
## tasksingles -0.563  0.489
## pas         -0.793  0.599  0.567
## tskqdrplt:p  0.522 -0.894 -0.433 -0.658
## tsksngls:ps  0.534 -0.459 -0.901 -0.673  0.506
```

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_experim
rt_partialpooling5 <- lmer(rt.obj ~ task*pas + (1|subject) + (1|trial) + (1|odd.digit), REML = FALSE, da
rt_partialpooling6 <- lmer(rt.obj ~ task*pas + (1|subject) + (1|trial) + (1|odd.digit) + (1|cue), REML =
rt_partialpooling7 <- lmer(rt.obj ~ task*pas + (1|subject) + (1|trial) + (1|odd.digit) + (1|cue) + (1|pa
rt_partialpooling8 <- lmer(rt.obj ~ task*pas + (1|subject) + (1|trial) + (1|odd.digit) + (1|cue) + (1|pa
rt_partialpooling9 <- lmer(rt.obj ~ task*pas + (1|subject) + (1|trial) + (1|odd.digit) + (1|cue) + (1|pa
## boundary (singular) fit: see ?isSingular
```

```
## Warning: Model failed to converge with 1 negative eigenvalue: -3.1e-02
```

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_partialpooling9), comp='Variance')
```

```
##  Groups     Name        Variance
##  trial      (Intercept) 0.00162119
##  cue        (Intercept) 0.08988376
##  seed       (Intercept) 0.08405683
##  subject    (Intercept) 0.01396917
##  even.digit (Intercept) 0.00000000
##  pas        (Intercept) 0.00222562
##  odd.digit  (Intercept) 0.00022564
##  Residual               8.11497710
```

```
?isSingular
```

```
## starting httpd help server ... done
```

The fit is singular because "even.digit" explains no variance, i.e. is estimated at exactly 0.

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

If you add two variables that explain exactly the same variance, you would end up with a singular fit. So optimally you want to add terms that explain a lot of variance of which no other variables explain.

Exercise 3

```
# 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

```r
pasmodel <- glmer(count ~ pas*task + (pas|subject), data = data.count, family = poisson, control = glme
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
##   4685.1   4719.6  -2333.6   4667.1      331
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.7718 -1.9208 -0.1275  1.6133 11.6477
##
## Random effects:
##  Groups  Name        Variance Std.Dev. Corr
##  subject (Intercept) 1.2017   1.0962
##          pas         0.2203   0.4694   -0.99
## Number of obs: 340, groups:  subject, 29
##
## Fixed effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)           4.28937    0.20577  20.846  < 2e-16 ***
## pas                  -0.19479    0.08798  -2.214   0.0268 *
## taskquadruplet        0.16669    0.04007   4.160 3.18e-05 ***
## tasksingles          -0.39660    0.04192  -9.461  < 2e-16 ***
## pas:taskquadruplet -0.07195    0.01606  -4.480 7.47e-06 ***
## pas:tasksingles       0.16855    0.01587  10.622  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) pas    tskqdr tsksng ps:tskq
## pas         -0.989
## taskqudrplt -0.100  0.083
## tasksingles -0.096  0.078  0.490
## ps:tskqdrpl  0.088 -0.091 -0.891 -0.430
## ps:tsksngls  0.089 -0.091 -0.456 -0.900  0.501
```

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

Poisson, because it's good for modelling frequency (count).

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 kind of 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, now it works :)

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

Did not have a converging fit, but will do so anyway.

```
pasmodel2 <- glmer(count ~ pas + task + (pas|subject), data = data.count, family = poisson)
summary(pasmodel2)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: count ~ pas + task + (pas | subject)
##    Data: data.count
##
##      AIC      BIC   logLik deviance df.resid
##   4923.2   4950.0  -2454.6   4909.2      333
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.9891 -2.1490 -0.1911  1.8234 11.0534
##
## Random effects:
##  Groups  Name        Variance Std.Dev. Corr
##  subject (Intercept) 1.2147   1.1022
##          pas         0.2232   0.4725   -0.99
## Number of obs: 340, groups:  subject, 29
##
## Fixed effects:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     4.213653   0.205804  20.474   <2e-16 ***
## pas            -0.162485   0.088064  -1.845    0.065 .
## taskquadruplet  0.006593   0.018184   0.363    0.717
## tasksingles     0.004307   0.018189   0.237    0.813
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) pas    tskqdr
## pas         -0.989
## taskqudrplt -0.044  0.000
## tasksingles -0.044  0.000  0.501
```

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

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

I would choose the model with the interaction effect included. This seems a too important distinction in the data to be ignored.

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*

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

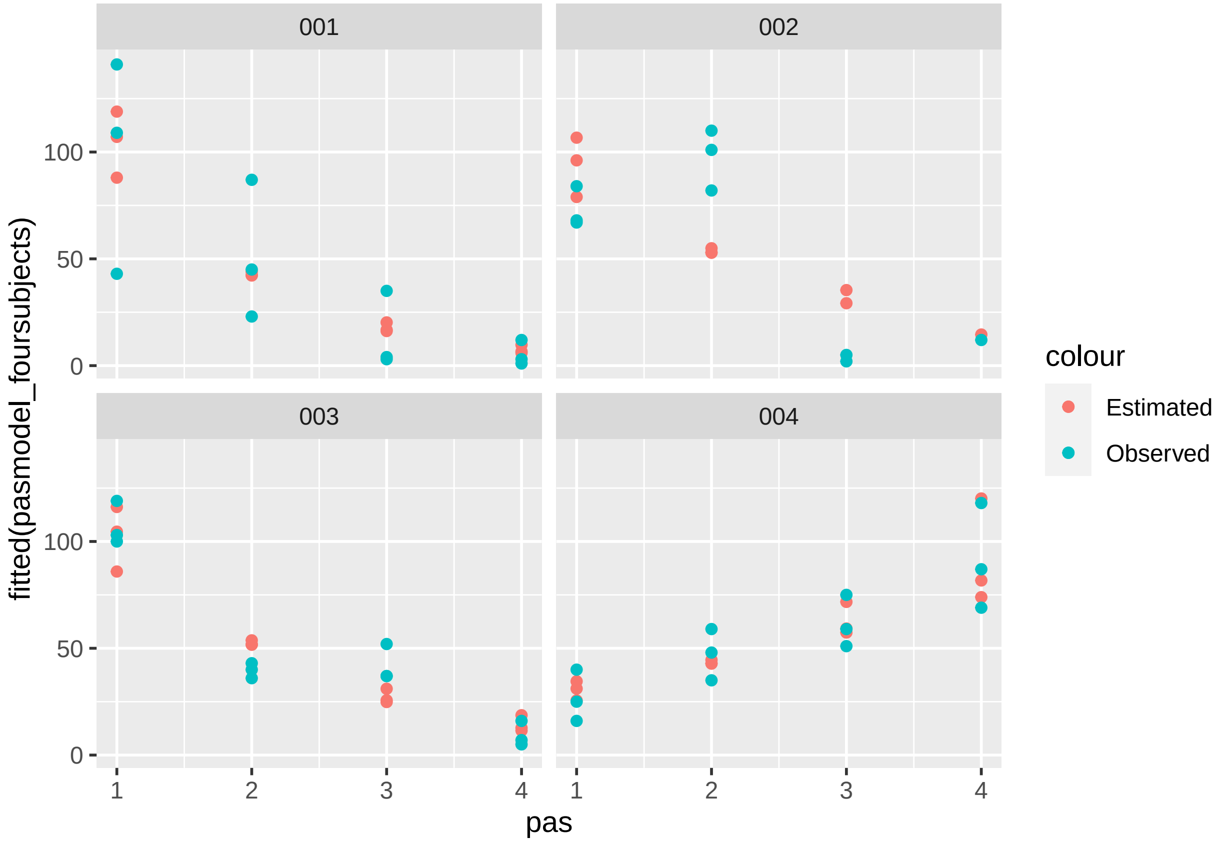
```
## # A tibble: 1 x 2
##   ‘SSR interaction model‘ ‘SSR no interaction‘
##                     <dbl>                <dbl>
## 1                   2508.                2749.
```

```
AIC(pasmodel, pasmodel2)
```

```
##           df      AIC
## pasmodel   9 4685.119
## pasmodel2  7 4923.190
```

3. 2. vii.

```
# 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 = poiss

# 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
##  17425.0  17464.0  -8707.5  17415.0    18126
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -8.1096 -0.6101  0.3181  0.5653  1.6476
##
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.2004   0.4477
## Number of obs: 18131, groups:  subject, 29
##
## Fixed effects:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -0.950104   0.098399  -9.656   <2e-16 ***
## taskquadruplet -0.029418   0.045016  -0.653    0.513
## tasksingles    -0.008914   0.046889  -0.190    0.849
## pas             1.014031   0.022900  44.281   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr tsksng
## taskqudrplt -0.247
## tasksingles -0.189  0.489
## pas         -0.421  0.030 -0.083
```

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
##  17421.5  17444.9  -8707.7  17415.5    18128
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -8.1864 -0.6117  0.3187  0.5664  1.6348
##
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.2005   0.4478
## Number of obs: 18131, groups:  subject, 29
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.96488    0.09504  -10.15   <2e-16 ***
## pas          1.01488    0.02275   44.62   <2e-16 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Correlation of Fixed Effects:
##     (Intr)
## pas -0.440
```

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
## -2.4709  -1.2372   0.4923   0.7704   1.1188
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.70591    0.07006 -10.077   <2e-16 ***
## taskquadruplet      0.05152    0.09696   0.531    0.595
## tasksingles        -0.10956    0.10248  -1.069    0.285
## pas                 0.88428    0.03526  25.081   <2e-16 ***
## taskquadruplet:pas -0.04835    0.04925  -0.982    0.326
## tasksingles:pas     0.07069    0.05017   1.409    0.159
## ---
## 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: 17864  on 18125  degrees of freedom
## AIC: 17876
##
## Number of Fisher Scoring iterations: 5
```

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

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.