practical_exercise_5, Methods 3, 2021, autumn semester

SIRID WIHLBORG

27/10/21

Exercises and objectives

The objectives of the exercises of this assignment are based on: https://doi.org/10.1016/j.concog.2019.03.007

4) Download and organise the data from experiment 1

5) Use log-likelihood ratio tests to evaluate logistic regression models

6) Test linear hypotheses

7) Estimate psychometric functions for the Perceptual Awareness Scale and evaluate them

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 is part 2 of Assignment 2 and will be part of your final portfolio

EXERCISE 4 - Download and organise the data from experiment 1

Go to https://osf.io/ecxsj/files/ and download the files associated with Experiment 1 (there should be 29). The data is associated with Experiment 1 of the article at the following DOI https://doi.org/10.1016/j.concog.2019.03.007

1) Put the data from all subjects into a single data frame - note that some of the subjects do not have the *seed* variable. For these subjects, add this variable and make in *NA* for all observations. (The *seed* variable will not be part of the analysis and is not an experimental variable)
    i. Factorise the variables that need factorising
    ii. Remove the practice trials from the dataset (see the *trial.type* variable)
    iii. Create a *correct* variable
    iv. Describe how the *target.contrast* and *target.frames* variables differ compared to the data from part 1 of this assignment

```
list <- list.files(path = "data/experiment_1", pattern = "*.csv", full.names=TRUE) # importing all files
df <- ldply(list, read_csv) # making them into one data-frame
```

```
## Rows: 882 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.

## Rows: 882 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.

## Rows: 882 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.

## Rows: 882 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.

## Rows: 882 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.

## Rows: 882 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.

## Rows: 882 Columns: 16

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): trial.type, task, target.type, obj.resp, subject
## dbl (11): 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.

## Rows: 882 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.

## Rows: 882 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.

## Rows: 882 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.

## Rows: 860 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.

## Rows: 882 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.

## Rows: 882 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.

## Rows: 882 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.
```

```
## Rows: 882 Columns: 16

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): trial.type, task, target.type, obj.resp, subject
## dbl (11): 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.

## Rows: 882 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.

## Rows: 882 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.

## Rows: 882 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.

## Rows: 882 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.

## Rows: 882 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.

## Rows: 882 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.

## Rows: 882 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.

## Rows: 882 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.

## Rows: 910 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.

## Rows: 900 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.

## Rows: 882 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.

## Rows: 882 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.

## Rows: 882 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.
```

```
## Rows: 882 Columns: 16

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): trial.type, task, target.type, obj.resp, subject
## dbl (11): 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.
```

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

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

df <- df %>%
  mutate(subject = as.factor(subject)) %>%
  mutate(task = as.factor(task)) %>%
  mutate(cue = as.factor(cue)) %>%
  mutate(pas = as.factor(pas)) %>%
  mutate(trial = as.factor(trial)) %>%
  mutate(trial.type = as.factor(trial.type)) %>%
  mutate(target.frames = as.integer(target.frames))


unique(df$target.contrast)
```

```
## [1] 0.1
```

```
unique(df$target.frames)
```

```
## [1] 3 2 1 5 6 4
```

We see that "target.contrast" only has one value (0.1) whereas in the previous assignment this variable had multiple values. "target.frames" has now 6 values (1:6), whereas in experiment 2 it was a fixed value (3), this means that in this experiment the experimenters varied the *duration* of the cue rather than the *contrast*.

EXERCISE 5 - Use log-likelihood ratio tests to evaluate logistic regression models

1) Do logistic regression - *correct* as the dependent variable and *target.frames* as the independent variable. (Make sure that you understand what *target.frames* encode). Create two models - a pooled model and a partial-pooling model. The partial-pooling model should include a subject-specific intercept.

```
m_pool <- glm(correct ~ target.frames, data = df, family = binomial) # complete pooling model
m_partial_pool <- glmer(correct ~ target.frames + (1|subject), data = df, family = binomial) # partial
```

i. the likelihood-function for logistic regression is: $L(p)={\displaystyle\prod_{i=1}^Np^{y_i}(1-p)^{(

```
likelihood_function <- function(model, y) {
  p <- fitted(model)
  y <- y

  return(prod(p^y*(1-p)^(1-y)))
}
```

ii. the log-likelihood-function for logistic regression is: $l(p) = {\displaystyle\sum_{i=1}^N}[y_i\ln{

```
log_likelihood_function <- function(model, y) {
  p <- fitted(model)
  y <- y

  return(sum(y*log(p)+(1-y)*log(1-p)))
}
```

iii. apply both functions to the pooling model you just created. Make sure that the log-likelihood matc

```
likelihood_function(m_pool, df$correct)
```

```
## [1] 0
```

```
log_likelihood_function(m_pool, df$correct)
```

```
## [1] -10865.25
```

```
logLik(m_pool)
```

```
## 'log Lik.' -10865.25 (df=2)
```

The Log likelihood value generated by my function matches the output from "logLik"-function in R.

The output of the likelihood function is 0 which at first seems surprising. However, since I'm multiplying a lot of values between 0-1 I do expect to get a very very small value. So if the computer has limited precision and the data-set is beg enough, the likelihood won't be of any use. In that case we can luckily turn to the log-likelihood.

iv. now show that the log-likelihood is a little off when applied to the partial pooling model - (the li

```
logLik(m_partial_pool)
```

```
## 'log Lik.' -10622.03 (df=3)
```

```
log_likelihood_function(m_partial_pool, df$correct)
```

```
## [1] -10565.53
```

The loglikelihood value from "logLike" function is -10622.03 and from our function we get -10565.53, so yes these are not completely similar.

2) Use log-likelihood ratio tests to argue for the addition of predictor variables, start from the null model, `glm(correct ~ 1, 'binomial', data)`, then add subject-level intercepts, then add a group-level effect of *target.frames* and finally add subject-level slopes for *target.frames*. Also assess whether or not a correlation between the subject-level slopes and the subject-level intercepts should be included.

```
m_null <-          glm(correct ~ 1, family = binomial, data = df)
m_ran.int <-       glmer(correct ~ 1 + (1|subject), family = binomial, data = df)
m_target.int <-    glmer(correct ~ target.frames + (1|subject), family = binomial, data = df)
m_target.slope <-  glmer(correct ~ target.frames + (target.frames|subject), family = binomial, data = d
m_target.slope.no.cor <- glmer(correct ~ target.frames + (target.frames||subject), family = binomial, da

model <- c("m_null", "m_ran.int", "m_target.int", "m_target.slope.no.cor", "m_target.slope")
logLik_values <- anova(m_ran.int, m_null, m_target.int, m_target.slope, m_target.slope.no.cor)$logLik

as.tibble(cbind(model, logLik_values))
```

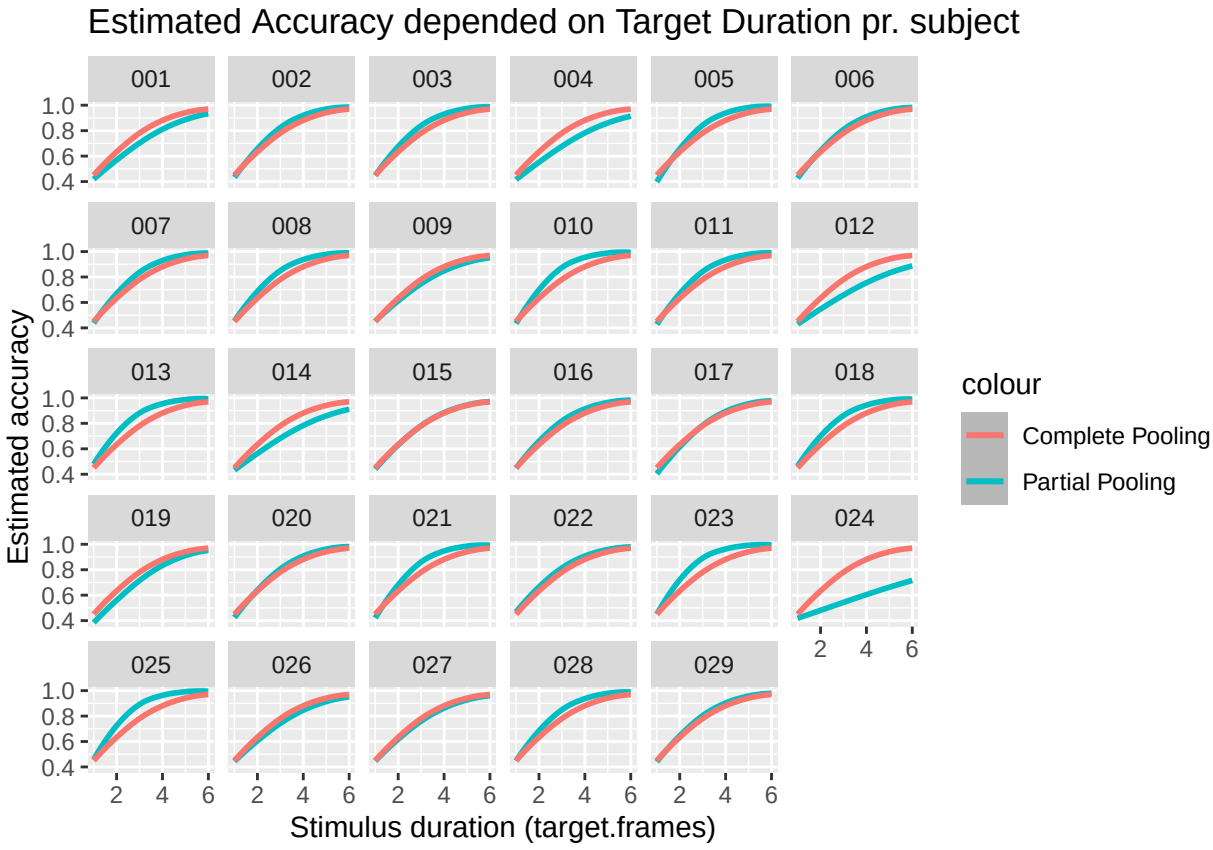
```
## Warning: 'as.tibble()' was deprecated in tibble 2.0.0.
## Please use 'as_tibble()' instead.
## The signature and semantics have changed, see '?as_tibble'.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_warnings()' to see where this warning was generated.
```

```
## # A tibble: 5 x 2
##   model                 logLik_values
##   <chr>                 <chr>
## 1 m_null                -13341.5389675897
## 2 m_ran.int             -13157.5490854926
## 3 m_target.int          -10622.0314964961
## 4 m_target.slope.no.cor -10460.2881343787
## 5 m_target.slope        -10448.8251446113
```

I see that adding target.frames as a fixed effect and including target.frames as random slope and subject as random intercept whilst including their correlation creates the LogLikelihood value closest to zero ie. is the best model.

```
i. write a short methods section and a results section where you indicate which model you chose and the
ii. also include in the results section whether the fit didn't look good for any of the subjects. If so
```

```
df %>%
  ggplot() +
   geom_smooth(aes(x = target.frames, y = fitted(m_target.slope), color = "Partial Pooling")) +
   geom_smooth(aes(x = target.frames, y = fitted(m_pool), color = "Complete Pooling")) +
   facet_wrap( ~ subject) +
   labs(title = "Estimated Accuracy depended on Target Duration pr. subject") +
   labs(x = "Stimulus duration (target.frames)", y = "Estimated accuracy")
```




```
df_024 <- df %>%
  filter(subject == "024")

mean(df_024$correct)
```

```
## [1] 0.5675057
```

```
t.test((df_024$correct), mu=0.5) # checking if the accuracy of participant 024 is significantly differe
```

```
##
##  One Sample t-test
##
## data:  (df_024$correct)
## t = 4.026, df = 873, p-value = 6.167e-05
## alternative hypothesis: true mean is not equal to 0.5
## 95 percent confidence interval:
##  0.5345964 0.6004150
## sample estimates:
## mean of x
## 0.5675057
```

Using likelihood ratio tests I compared models that included target.frames as fixed effect to intercept models whilst varying the random effects. I found that the model that best predicted accuracy had target.frames

as fixed effect and for random effects; subject-level for random intercept and target.frames for random slope and including their interaction (

$$\chi^2(1) = 22.93, p < 0.00001$$

).

I've plotted the estimated group-level function pr. subject against a function that's based on a pooled model that hence generates a "grand-slope". I see that the functions generally fit together except for participant 024 that generally has low accuracy However, the performance for this participant was significantly larger than chance (>50%) which was validated using a one-sample t-test.

3) Now add *pas* to the group-level effects - if a log-likelihood ratio test justifies this, also add the interaction between *pas* and *target.frames* and check whether a log-likelihood ratio test justifies this

```
m_target.pas <- glmer(correct ~ target.frames + pas + (target.frames|subject), family = binomial, data =
m_target.pas.interact <- glmer(correct ~ target.frames * pas + (target.frames|subject), family = binomia

model <- c("m_target.int", "m_target.pas", "m_target.pas.interact")
logLik <- c(anova(m_target.int, m_target.pas, m_target.pas.interact)$logLik)
as.tibble(cbind(model, logLik))
```

```
## # A tibble: 3 x 2
##   model                 logLik
##   <chr>                 <chr>
## 1 m_target.int          -10622.0314964961
## 2 m_target.pas          -9931.8284235453
## 3 m_target.pas.interact -9742.03938109804
```

Using log-likelihood ratio test I compared three models with varying fixed effects. One with target.frames and PAS as fixed effect, another similar model but this time including the interaction between PAS and target.frames and lastly the model with only target.frames as fixed effect.

I found that the model including both predictors and their interaction is significantly a better model (

$$\chi^2(3) = 379.58, p < 0.00001$$

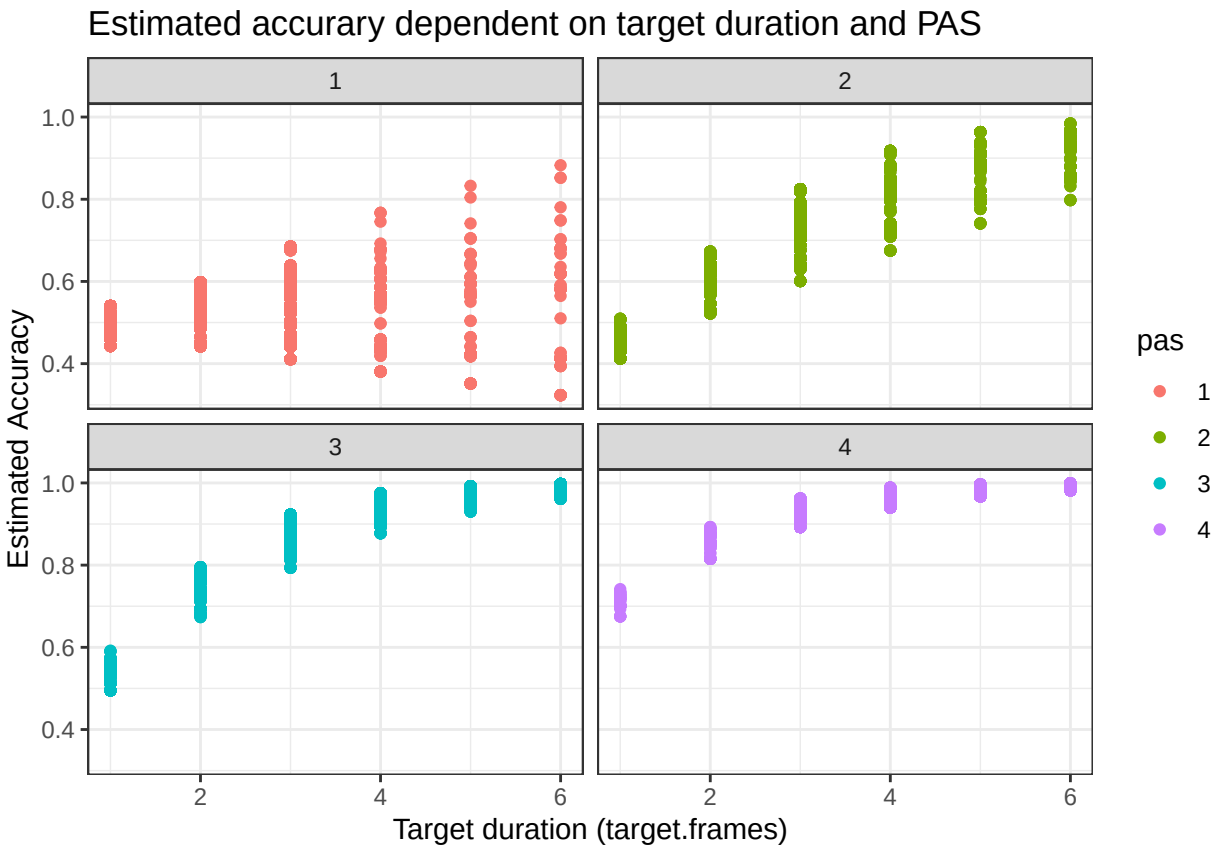
).

i. if your model doesn't converge, try a different optimizer

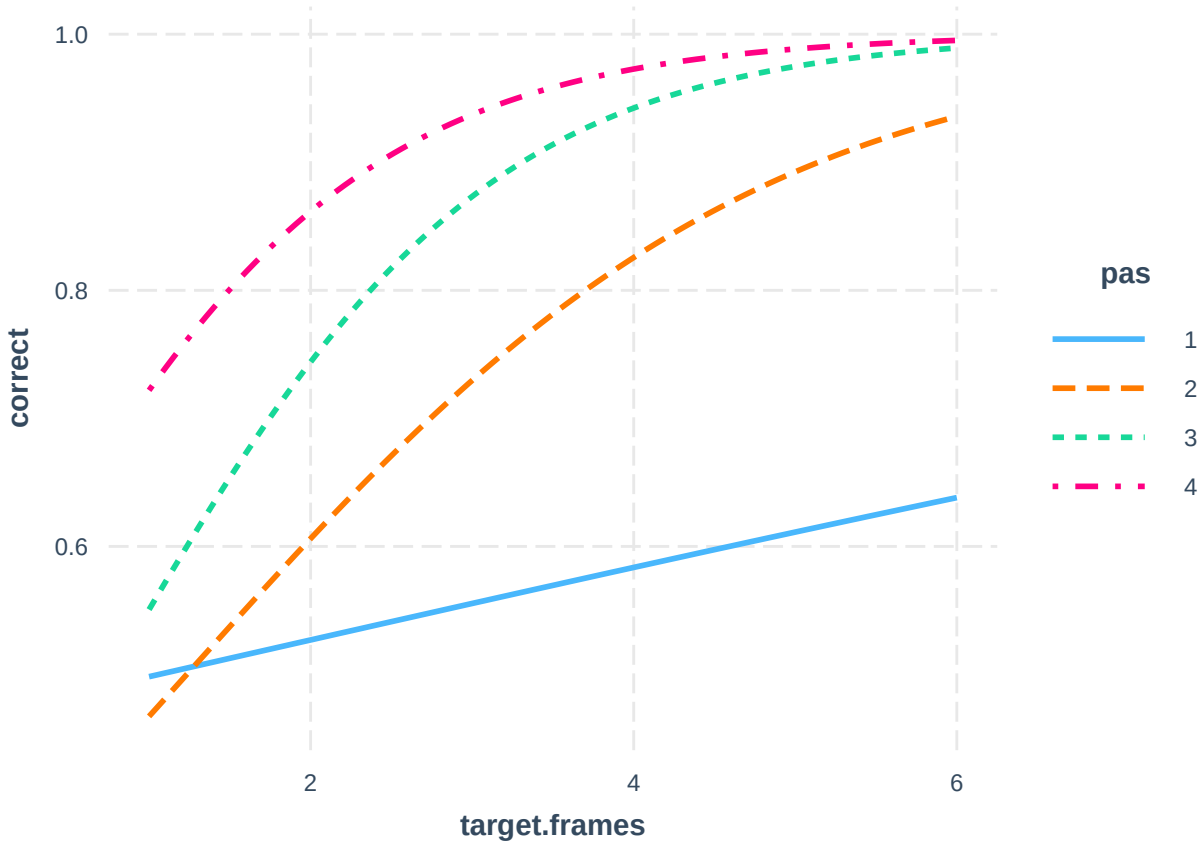
It did converge.

ii. plot the estimated group-level functions over 'xlim=c(0, 8)' for each of the four PAS-ratings - add

```
# Plot showing accuracy dependend on target.duration and PAS pr. PAS
df %>%
  ggplot() +
  geom_point(aes(x = target.frames, y = fitted(m_target.pas.interact), color = pas))+
  facet_wrap( ~ pas) +
  labs(title = "Estimated accurary dependent on target duration and PAS") +
  labs(x = "Target duration (target.frames)", y = "Estimated Accuracy") +
  theme_bw()
```




```
# Similar plot but more intuitive
interactions::interact_plot(model = m_target.pas.interact, pred = "target.frames", modx = "pas") # visu
```




```
# creating a table showing the percentile increase in accuracy for every fixed effect and interactions
estimates <- c(coef(summary(m_target.pas.interact))[1:8])
accuracy_increase_in_prob <- c(invlogit(estimates))
estimates <- c("intercept", "target.frames", "pas2", "pas3", "pas4", "target.frames:pas2", "target.frame

as.tibble(cbind(estimates, accuracy_increase_in_prob))
```

```
## # A tibble: 8 x 2
##   estimates          accuracy_increase_in_prob
##   <chr>              <chr>
## 1 intercept          0.469627661225394
## 2 target.frames      0.528669077821521
## 3 pas2               0.360913808395008
## 4 pas3               0.368543606525442
## 5 pas4               0.550216621181487
## 6 target.frames:pas2 0.609970337596386
## 7 target.frames:pas3 0.678891697140287
## 8 target.frames:pas4 0.68119723555509
```

From the two plots I can quickly see that increase in target duration and PAS will increase accuracy. There's a big difference when going from pas1 to the other ratings on the pas-scales.

Having pas1 as baseline there's a 36% increase in accuracy going to pas2, 37% increase to pas3 and 55% increase when going to pas4. The effect of target.frames is 60% increased on accuracy when pas changes from 1 to 2. The effect of target.frames is 68% increased on accuracy when pas changes from 1 to 3 and from 1 to 4.

We see an 47% accuracy when target.frames is 0 (baseline value) which means that performance drop to *under* chance level when target-duration is 0. This will be commented on and dealt with later in the assignment.

EXERCISE 6 - Test linear hypotheses

In this section we are going to test different hypotheses. We assume that we have already proved that more objective evidence (longer duration of stimuli) is sufficient to increase accuracy in and of itself and that more subjective evidence (higher PAS ratings) is also sufficient to increase accuracy in and of itself.
We want to test a hypothesis for each of the three neighbouring differences in PAS, i.e. the difference between 2 and 1, the difference between 3 and 2 and the difference between 4 and 3. More specifically, we want to test the hypothesis that accuracy increases faster with objective evidence if subjective evidence is higher at the same time, i.e. we want to test for an interaction.

1) Fit a model based on the following formula: `correct ~ pas * target.frames + (target.frames | subject))`
    i. First, use `summary` (yes, you are allowed to!) to argue that accuracy increases faster with objective evidence for PAS 2 than for PAS 1.

This is done in 5.2.ii.

2) `summary` won't allow you to test whether accuracy increases faster with objective evidence for PAS 3 than for PAS 2 (unless you use `relevel`, which you are not allowed to in this exercise). Instead, we'll be using the function `glht` from the `multcomp` package
    i. To redo the test in 6.1.i, you can create a *contrast* vector. This vector will have the length of the number of estimated group-level effects and any specific contrast you can think of can be specified using this. For redoing the test from 6.1.i, the code snippet below will do
    ii. Now test the hypothesis that accuracy increases faster with objective evidence for PAS 3 than for PAS 2.
    iii. Also test the hypothesis that accuracy increases faster with objective evidence for PAS 4 than for PAS 3
3) Finally, test that whether the difference between PAS 2 and 1 (tested in 6.1.i) is greater than the difference between PAS 4 and 3 (tested in 6.2.iii)

Snippet for 6.2.i

```
## testing whether PAS 2 is different from PAS 1
contrast.vector <- matrix(c(0, 0, 0, 0, 0, 1, 0, 0), nrow=1)
gh_1 <- glht(m_target.pas.interact, contrast.vector)
print(summary(gh_1))
invlogit(coef(summary(gh_1))) # finding increase in percentile

## as another example, we could also test whether there is a difference in
## intercepts between PAS 2 and PAS 3
contrast.vector <- matrix(c(0, -1, 1, 0, 0, 0, 0, 0), nrow=1)
gh_2 <- glht(m_target.pas.interact, contrast.vector)
print(summary(gh_2))
```

```
# testing if accuracy performance increases faster for pas3 than pas2
contrast.vector <- matrix(c(0, -1, 0, 1, 0, 0, 0, 0), nrow=1)
gh_3 <- glht(m_target.pas.interact, contrast.vector)
print(summary(gh_3))
invlogit(coef(summary(gh_3)))

# testing if accuracy performance increases faster for pas4 than pas3
contrast.vector <- matrix(c(0, -1, 0, 0, 1, 0, 0, 0), nrow=1)
gh_4 <- glht(m_target.pas.interact, contrast.vector)
print(summary(gh_4))
invlogit(coef(summary(gh_4)))
```

We see that there's a 34% increase in accuracy when going from pas2 to pas3 hence we fail to reject the null-hypothesis that there's not a faster increase in accuracy when going from pas2 to pas3.

There's a 52% increase when going from pas3 to pas4.

Since the increase from pas1 to pas2 is 60.9% and only 52% from pas3 to pas4, I conclude that the first increase is greater.

EXERCISE 7 - Estimate psychometric functions for the Perceptual Awareness Scale and evaluate them

We saw in 5.3 that the estimated functions went below chance at a target duration of 0 frames (0 ms). This does not seem reasonable, so we will be trying a different approach for fitting here.
We will fit the following function that results in a sigmoid, $f(x) = a + \frac{b-a}{1+e^{\frac{c-x}{d}}}$

It has four parameters: a, which can be interpreted as the minimum accuracy level, b, which can be interpreted as the maximum accuracy level, c, which can be interpreted as the so-called inflexion point, i.e. where the derivative of the sigmoid reaches its maximum and d, which can be interpreted as the steepness at the inflexion point. (When d goes towards infinity, the slope goes towards a straight line, and when it goes towards 0, the slope goes towards a step function).

We can define a function of a residual sum of squares as below

```
RSS <- function(dataset, par)
{
    ## "dataset" should be a data.frame containing the variables x (target.frames)
    ## and y (correct)

    ## "par" are our four parameters (a numeric vector)
    a = par[1]
    b = par[2]
    c = par[3]
    d = par[4]

    x <- dataset$x
    y <- dataset$y

    y.hat <- a + ((b-a)/(1+exp(1)^((c-x)/d)))

    RSS <- sum((y - y.hat)^2)
    return(RSS)
}
```

1) Now, we will fit the sigmoid for the four PAS ratings for Subject 7

   i. use the function `optim`. It returns a list that among other things contains the four estimated parameters. You should set the following arguments:
   `par`: you can set c and d as 1. Find good choices for a and b yourself (and argue why they are appropriate)
   `fn`: which function to minimise?
   `data`: the data frame with x, *target.frames*, and y, *correct* in it
   `method`: 'L-BFGS-B'
   `lower`: lower bounds for the four parameters, (the lowest value they can take), you can set c and d as `-Inf`. Find good choices for a and b yourself (and argue why they are appropriate)
   `upper`: upper bounds for the four parameters, (the highest value they can take) can set c and d as `Inf`. Find good choices for a and b yourself (and argue why they are appropriate)

```
df_007 <- df %>%
  dplyr::filter(subject == "007") %>%
  dplyr::select(target.frames, correct) %>%
  dplyr::rename(x = target.frames, y = correct)

par <- c(0.5, 0.8, 1, 1)

optim_007 <- optim(data = df_007, fn = RSS, par = par, method = 'L-BFGS-B', lower = c(0, 0, -Inf, -Inf)
```

Argument for par = (0.5, 0.8, 1, 1): I thought that the lowest accuracy-level (a) should be 50% since this is chance level. The most likely 'best accuracy' (b) is maybe 0.8, 80% seems accuracy reasonable. Since the function is sigmoid, the lower bound (lower =) is set to 0 and upper (upper =) to 1.

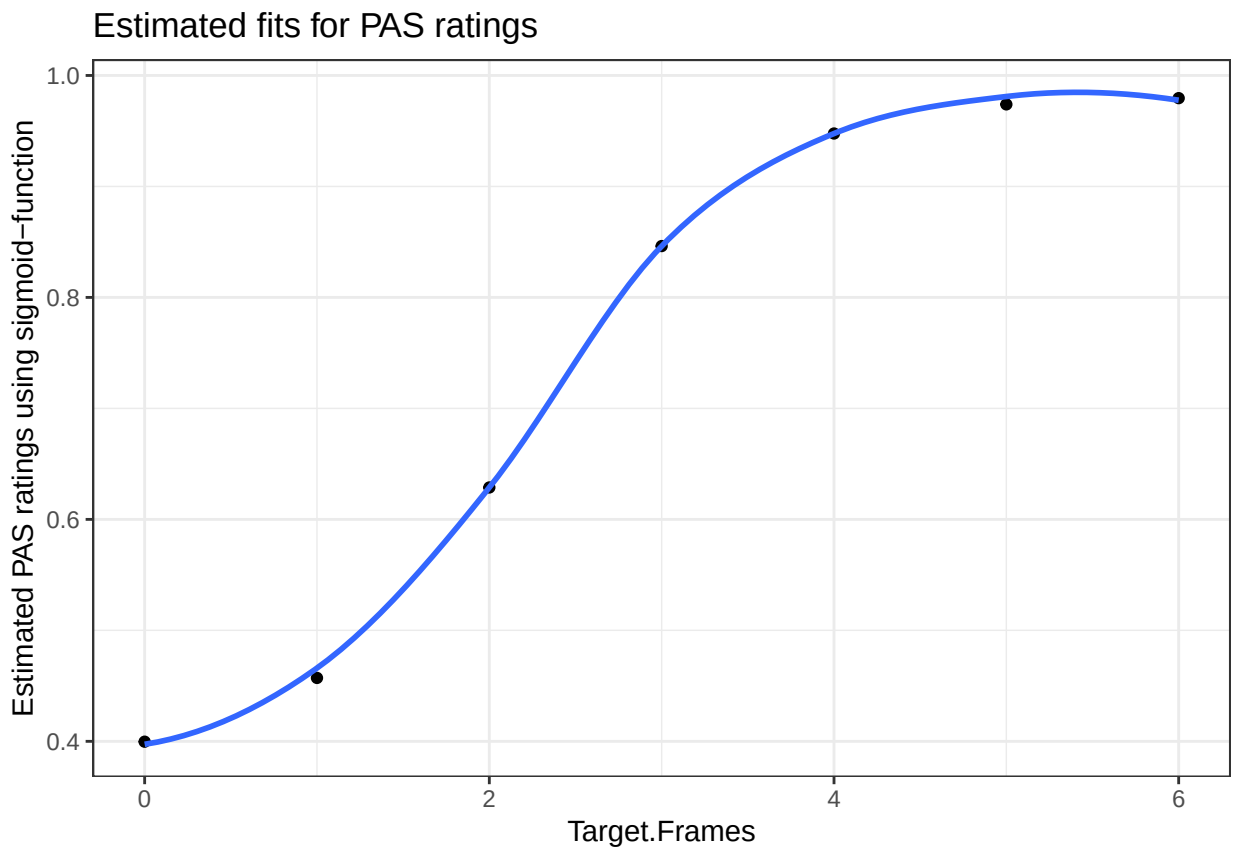
Results: We got the following suggestions for the estimated parameters. a = 0.38. b = 0.98. c = 2.23. d = 0.63.

Optim-function gives us a lowest accuracy on 38% and highest 98%. I would think it's a bit extreme, and doesn't really solve the problem of us wanting a 50% lowest accuracy-level. c = 2.23, which would mean that the function is most steep (ie. the highest increase in accuracy) when you go from pas 2 to pas 3. The steepness at the inflexion point is d = 0.63 #you said we should make the lower bound -Inf, but you say "when it goes towards 0", what is in fact the lower bound?#

ii. Plot the fits for the PAS ratings on a single plot (for subject 7) 'xlim=c(0, 8)'

```
# Using the estimated parameters suggested by the optim-function in the sigmoid-function for participan
sigmoid_function_007 <- function(x) {
  optim_007$par[1] + ((optim_007$par[2]-optim_007$par[1])/(1+exp(1)^((optim_007$par[3]-x)/optim_007$par

ggplot() +
  geom_point(aes(x = c(0:6), y = sigmoid_function_007(0:6))) +
  geom_smooth(aes(x = c(0:6), y = sigmoid_function_007(0:6)), se = FALSE) +
  labs(title = "Estimated fits for PAS ratings",
       x = "Target.Frames",
       y = "Estimated PAS ratings using sigmoid-function") +
  theme_bw()
```



iii. Create a similar plot for the PAS ratings on a single plot (for subject 7), but this time based on

```
# the model from 6.1: m_target.pas.interact = correct ~ target.frames * pas + (target.frames | subject)
```

iv. Comment on the differences between the fits - mention some advantages and disadvantages of each way

2) Finally, estimate the parameters for all subjects and each of their four PAS ratings. Then plot the estimated function at the group-level by taking the mean for each of the four parameters, a, b, c and d across subjects. A function should be estimated for each PAS-rating (it should look somewhat similar to Fig. 3 from the article: https://doi.org/10.1016/j.concog.2019.03.007)

   i. compare with the figure you made in 5.3.ii and comment on the differences between the fits - mention some advantages and disadvantages of both.