practical_exercise_5, Methods 3, 2021, autumn semester

Rikke Uldbæk

27/11 2021

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

- The target frame is the variable that determines how many frames the target is shown in, i.e for how long the target is shown. In experiment 1 the target frame ranges from 1-6 and in experiment 2 the target frame is 3 throughout the entire experiment.

- The target contrast variable is a variable that represents the grey-scale proportion of the target digit. In experiment 1 the target contrast is a fixed number (01) for each participant throughout the entire experiment. In experiment 2 the target contrast is adjusted to fit each individual, i.e the each subject has a different target contrast.

```r
# read the data
df <- read_bulk(directory = "experiment_1/", fun = read_csv)

# Get rid of the seed variable
df <- df %>%
    dplyr::select(!seed)

# Create a _correct_ variable
df$correct <- (ifelse(df$obj.resp == "o" & df$target.type ==
    "odd", "1", ifelse(df$obj.resp == "e" & df$target.type ==
    "even", "1", 0)))

# Factorize the variables that need factorising
df$trial.type <- as.factor(df$trial.type)
df$pas <- as.factor(df$pas)
df$trial <- as.factor(df$trial)
df$cue <- as.factor(df$cue)
df$task <- as.factor(df$task)
df$obj.resp <- as.factor(df$obj.resp)
df$subject <- as.factor(df$subject)
df$target.frames <- as.integer(df$target.frames)


# Remove the practice trials from the dataset (see the
# _trial.type_ variable)
df <- df %>%
    subset(trial.type == "experiment")
```

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

Exercise 5.1

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.

i.Create likelihood function

The likelihood-function for logistic regression is: $L(p) = \prod_{i=1}^{N} p^{y_i}(1-p)^{(1-y_i)}$ **(Remember the probability mass function for the Bernoulli Distribution). Create a function that calculates the likelihood.** See chunk.

ii. Create log-likelihood function

The log-likelihood-function for logistic regression is: $l(p) = \sum_{i=1}^{N} [y_i \ln p + (1 - y_i) \ln (1 - p)$. **Create a function that calculates the log-likelihood** See chunk

iii. Applying the functions

Apply both functions to the pooling model you just created. Make sure that the log-likelihood matches what is returned from the *logLik* function for the pooled model.Does the likelihood-function return a value that is surprising? Why is the log-likelihood preferable when working with computers with limited precision? The output of the loglik-function matches my log-likelihood function, they are both -10865.25. Considering the size of the data set, this value is not so surprising, because the likelihood for observing the exact same data in the exact same order given our model is fairly small (and the loglik is super small too). When using log-likelihood function we avoid getting really small numbers (eg. 10.000's of 0's in a decimal number), so doing this, makes it easier for computers with limited precision to cope with the numbers.

iv. log-likelihood and partial pooling

Now show that the log-likelihood is a little off when applied to the partial pooling model - (the likelihood function is different for the multilevel function - see section 2.1 of https://www.researchgate.net/profile/Douglas-Bates/publication/2753537_Computational_Methods_for_Multilevel_Modelling/links/00b4953b4108d73427000000/Computational-Methods-for-Multilevel-Modelling.pdf if you are interested) From the comparison of my loglikelihood-function and the logLik-function it is evident that it's a little off, because the generated values are not the same. The LogLik function when applied on the partial pooling model = -10622.03 (df=3), and when my loglikelihood function is applied to the partial pooling model = -10565.53.

```r
# transform used variables (in order to make code run (the
# function))
df$correct <- as.numeric(df$correct)

# logistic regression with _correct_ as the dependent
# variable and _target.frames_ as the independent variable
# complete pooling
complete_pool_model <- glm(correct ~ target.frames, data = df,
    family = binomial(link = "logit"))


# partial-pooling model. The partial-pooling model should
# include a subject-specific intercept.
partial_pool_model <- glmer(correct ~ target.frames + (1 | subject),
    data = df, family = binomial(link = "logit"))


# Create a function that calculates the likelihood
likelihood <- function(model, y) {
    p <- fitted.values(model)
    return(prod(p^y * (1 - p)^(1 - y)))
}
likelihood(complete_pool_model, df$correct)  # = 0
```

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

```
# Create a function that calculates the log-likelihood
loglikelihood <- function(model, y) {
    p <- fitted.values(model)
    return(sum(y * log(p) + (1 - y) * log(1 - p)))
}
loglikelihood(complete_pool_model, df$correct)  # = -10865.25
```

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

```
# comparing with the logLike-function on complete pooling
# model
logLik(complete_pool_model)  # = -10865.25 (df=2)
```

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

```
# using the logLike-function and my log-likelihood-function
# for partial pooling model
logLik(partial_pool_model)  # = -10622.03 (df=3)
```

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

```
loglikelihood(partial_pool_model, df$correct)  # = -10565.53
```

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

Exercise 5.2

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. When comparing model 3 (with correlation) and model 4 (without correlation), it shows that model 3 has the best performance, thus I include a correlation between subject-level slopes and the subject-level intercepts.

i. Choosing the best model: Model 3

Write a short methods section and a results section where you indicate which model you chose and the statistics relevant for that choice. Include a plot of the estimated group-level function with `xlim=c(0, 8)` that includes the estimated subject-specific functions.

```
ii. also include in the results section whether the fit didn't look good for any of the subjects. If so
```

The model that performs the best is model 3; $(correct\ target.frames + (1 + target.frames|subject))$. This model predicts "correct" form "target.frames" as a fixed effect, "target.frames" as a random slope and "subject" as a random intercept, including their correlation. The model has the best performance with a $X^2 = 22,926, p < .01$.Model 3's $AIC = 20908$ and $logLik = -10449$, whilst being statistically significant, thus being the best model compared to the other 4 models.

The fit from model 3, did not look good for subject 1, 4, 9, 12, 14, and 24. From visual inspection it seems like, especially subject 24 performed differently from the other subjects. Subject 24's performance was worse than chance, as this persons function is located below the nullmodel's function. When comparing Subject 24 and mu in a t-test the results yields: $t(873) = 4.026, p < .01$, thus subjects 24's mean is significantly different from the mean of all participants in the negative direction (the performance is worse by chance).

```
# Use log-likelihood ratio tests to argue for the addition of
# predictor variables
nullmodel <- glm(correct ~ 1, data = df, family = binomial(link = "logit"))
model1 <- glmer(correct ~ 1 + (1 | subject), data = df, family = binomial(link = "logit"))
model2 <- glmer(correct ~ target.frames + (1 | subject), data = df,
    family = binomial(link = "logit"))
model3 <- glmer(correct ~ target.frames + (1 + target.frames |
    subject), data = df, family = binomial(link = "logit"))
model4 <- glmer(correct ~ target.frames + (1 + target.frames ||
    subject), data = df, family = binomial(link = "logit"))  # no correlation model

# Anova
anova(model1, model2, model3, model4, nullmodel)  #model 3
```

```
## Data: df
## Models:
## nullmodel: correct ~ 1
## model1: correct ~ 1 + (1 | subject)
## model2: correct ~ target.frames + (1 | subject)
## model4: correct ~ target.frames + (1 + target.frames || subject)
## model3: correct ~ target.frames + (1 + target.frames | subject)
##           npar   AIC   BIC logLik deviance    Chisq Df Pr(>Chisq)
## nullmodel    1 26685 26693 -13342    26683
## model1       2 26319 26335 -13158    26315  367.980  1  < 2.2e-16 ***
## model2       3 21250 21274 -10622    21244 5071.035  1  < 2.2e-16 ***
## model4       4 20929 20961 -10460    20921  323.487  1  < 2.2e-16 ***
## model3       5 20908 20948 -10449    20898   22.926  1  1.684e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

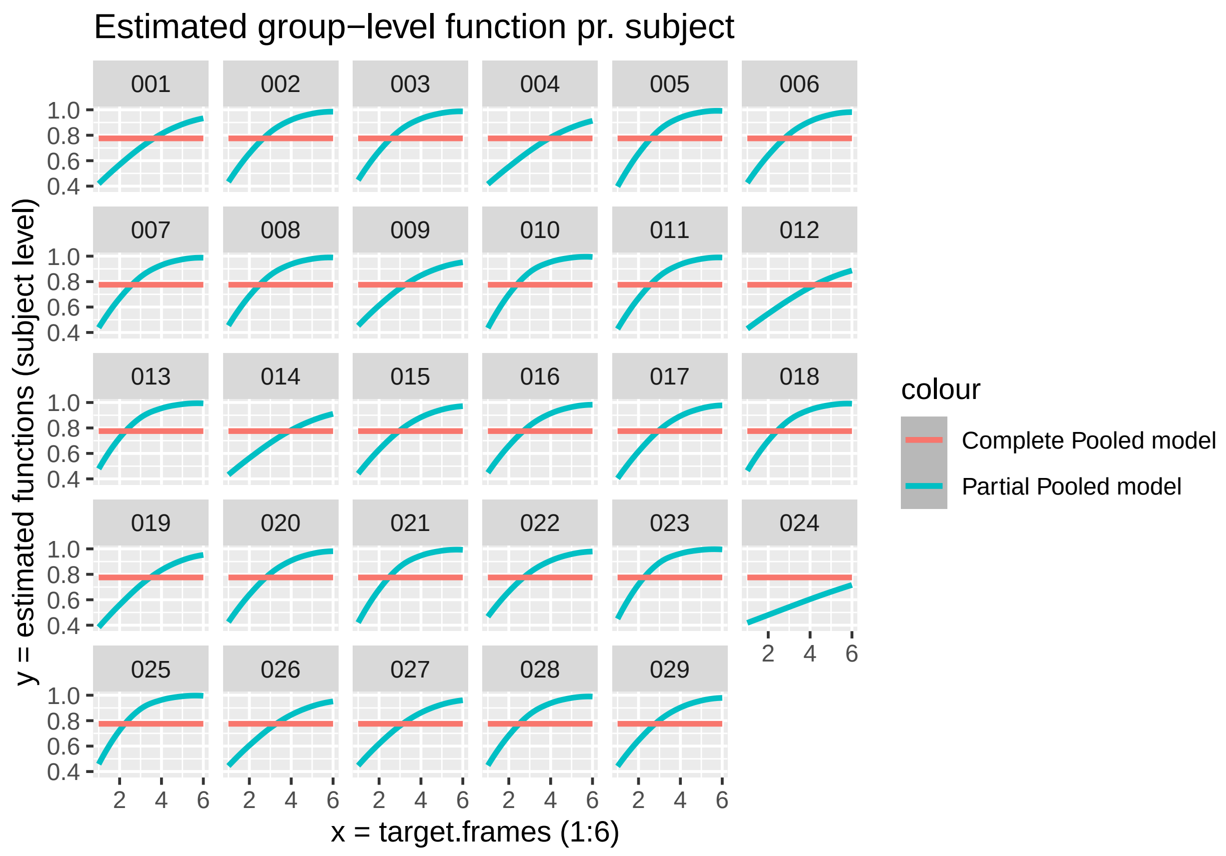
```
# Using the log-likelihood ratio test for correlation between
# the subject-level slopes and the subject-level intercepts
anova(model3, model4)  # model 3 is slightly better
```

```
## Data: df
## Models:
## model4: correct ~ target.frames + (1 + target.frames || subject)
## model3: correct ~ target.frames + (1 + target.frames | subject)
##        npar   AIC   BIC logLik deviance  Chisq Df Pr(>Chisq)
## model4    4 20929 20961 -10460    20921
## model3    5 20908 20948 -10449    20898 22.926  1  1.684e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# plot of the estimated group-level function with `xlim=c(0,
# 8)` that includes the estimated subject-specific functions.
# lav plot hvor vi sammenligner nullmodel med valgte model 3
```

```r
df %>%
    ggplot() + geom_smooth(aes(x = target.frames, y = fitted(model3),
    color = "Partial Pooled model")) + geom_smooth(aes(x = target.frames,
    y = fitted(nullmodel), color = "Complete Pooled model")) +
    facet_wrap(~subject) + labs(title = "Estimated group-level function pr. subject") +
    labs(x = "x = target.frames (1:6)", y = "y = estimated functions (subject level)")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



```r
# statistical test (50%) to see if subject 24 performs better
# than chance isolating subject 24
subject24 <- df %>%
    filter(subject == "024")


# making a t-test
t.test(subject24$correct, mu = 0.5, alternative = "greater")  #t-test tester for om den er significaly
```

```
##
##  One Sample t-test
##
## data:  subject24$correct
## t = 4.026, df = 873, p-value = 3.083e-05
## alternative hypothesis: true mean is greater than 0.5
## 95 percent confidence interval:
##  0.5398963       Inf
## sample estimates:
## mean of x
## 0.5675057
```

```r
t.test(subject24$correct, mu = 0.5, alternative = "less")  #t-test tester for om den er significaly dif
```

```
##
##  One Sample t-test
##
## data:  subject24$correct
## t = 4.026, df = 873, p-value = 1
## alternative hypothesis: true mean is less than 0.5
## 95 percent confidence interval:
##       -Inf 0.5951151
## sample estimates:
## mean of x
## 0.5675057
```

Exercise 5.3

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.

i. Adding "pas" and interaction between "pas" and "target.frames"

When comparing model 5 (including pas as a fixed effect) and model 6 (including pas as a fixed effect and the interaction between pas and target.frames), the log-likelihood ratio test justifies adding pas as a fixed effect and the interaction between pas and target.frames. Model 6 has a logLik value closer to 0 and a better AIC value than model 5.

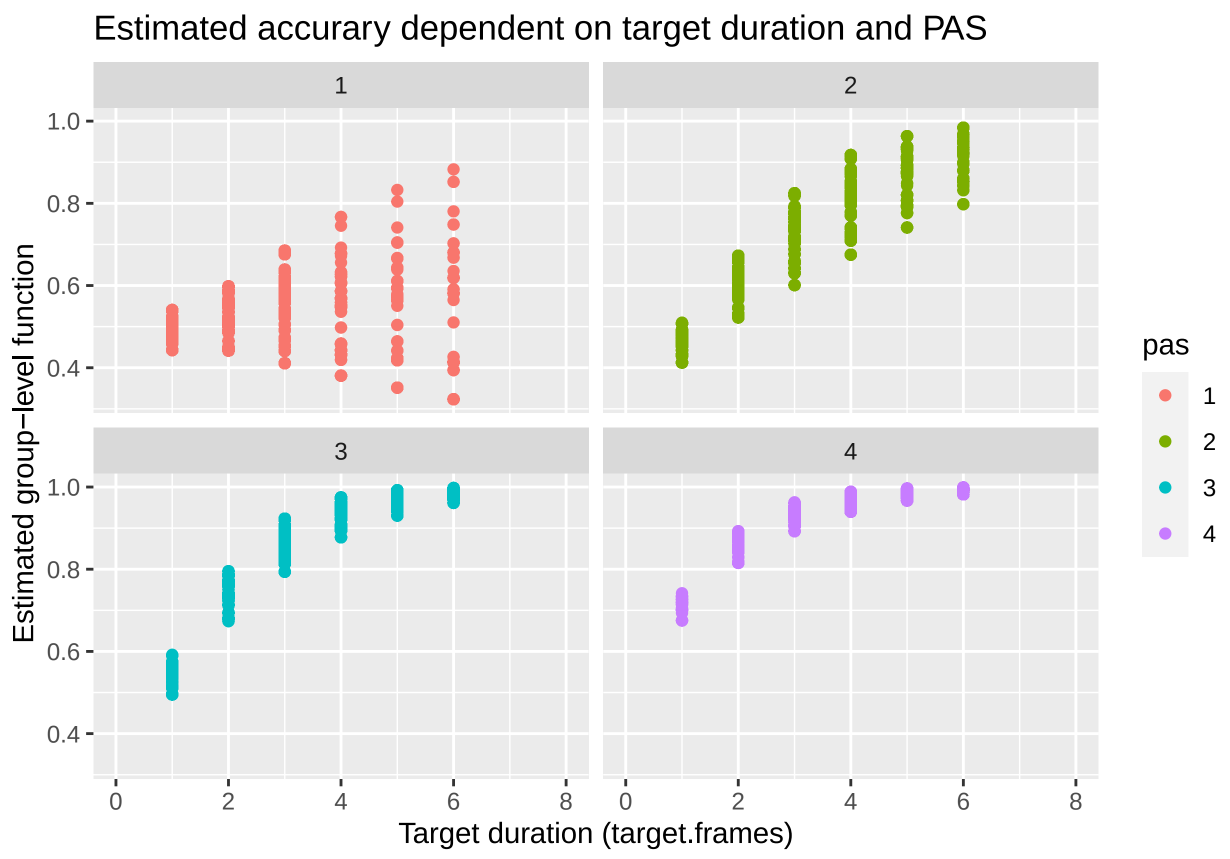
ii. Plotting estimated group level function for each "pas"-ratings

Plot the estimated group-level functions over `xlim=c(0, 8)` for each of the four PAS-ratings - add this plot to your report (see: 5.2.i) and add a description of your chosen model. Describe how *pas* affects accuracy together with target duration if at all. Also comment on the estimated functions' behaviour at target.frame=0 - is that behaviour reasonable? The plot shows how target duration affects accuracy (per increment of target frame(1-6) the accuracy increases), and how this is a general tendency within all 4 "pas". Per increment af "pas" (1-4) the baseline of accuracy increases, except for "pas" 2, here "pas" 1 has a higher baseline than "pas" 2.

```r
# adding pas as a fixed effect
model5 <- glmer(correct ~ target.frames + pas + (1 + target.frames |
    subject), data = df, family = binomial(link = "logit"), control = glmerControl(optimizer = "bobyqa")

model6 <- glmer(correct ~ target.frames + pas + target.frames:pas +
    (1 + target.frames | subject), data = df, family = binomial(link = "logit"),
    control = glmerControl(optimizer = "bobyqa"))  # adding pas and interaction between pas and target.

anova(model5, model6)  #model 6 is the best
```

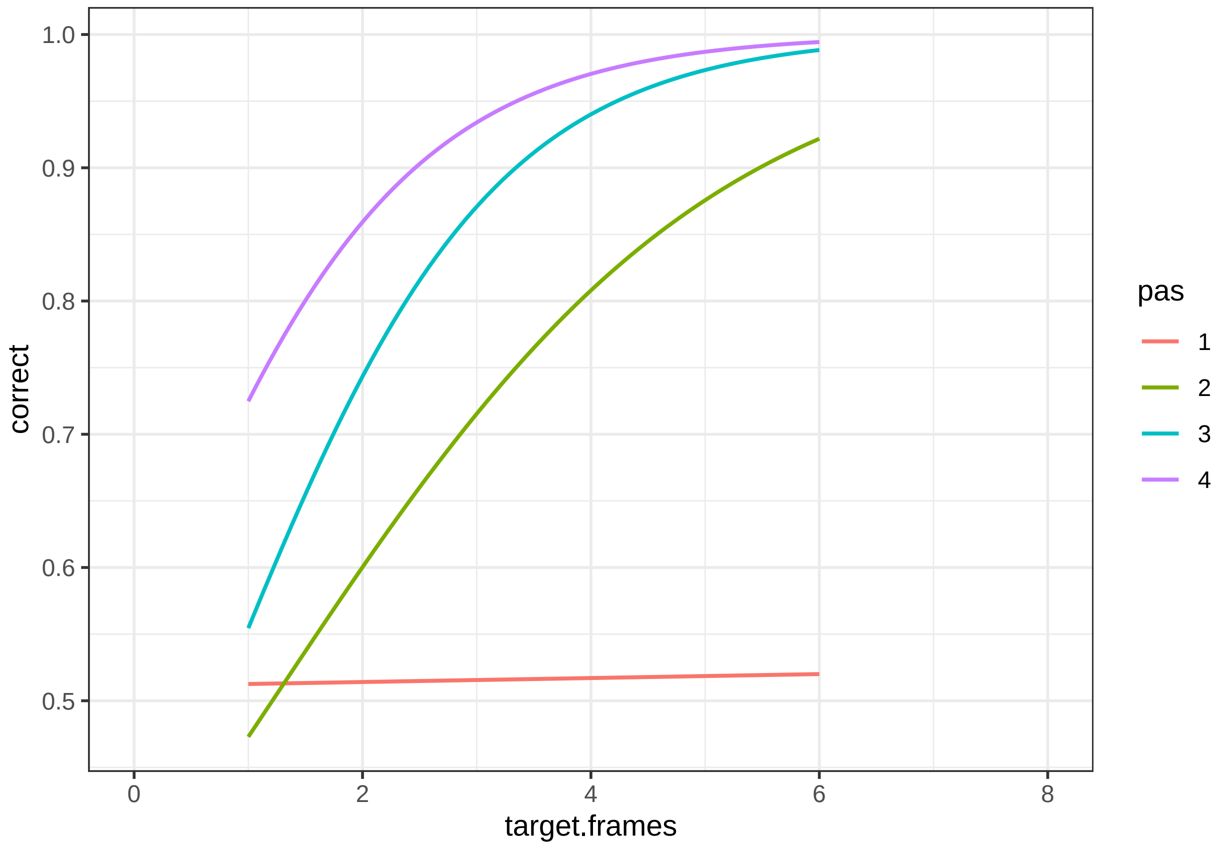
```
## Data: df
## Models:
## model5: correct ~ target.frames + pas + (1 + target.frames | subject)
## model6: correct ~ target.frames + pas + target.frames:pas + (1 + target.frames |
## model6:     subject)
##        npar   AIC   BIC  logLik deviance  Chisq Df Pr(>Chisq)
## model5    8 19880 19945 -9931.8    19864
## model6   11 19506 19596 -9742.0    19484 379.58  3  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Plot the estimated group-level functions over `xlim=c(0,
# 8)` for each of the four PAS-ratings
p <- df %>%
    ggplot() + geom_point((aes(x = target.frames, y = fitted(model6),
    color = pas))) + facet_wrap(. ~ pas) + labs(title = "Estimated accurary dependent on target duration
    labs(x = "Target duration (target.frames)", y = "Estimated group-level function")

p + xlim(0, 8)
```



```r
# another plot
ggplot(df, aes(x = target.frames, y = as.numeric(as.character(correct)),
    color = pas)) + geom_smooth(method = "glm", se = FALSE, method.args = list(family = "binomial"),
    size = 0.7) + xlim(min = 0, max = 8) + labs(y = "correct") +
    theme_bw()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



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.

Exercise 6.1

1) Fit a model based on the following formula: `correct ~ pas * target.frames + (target.frames | subject))`

i. Interpreting accuracy for “pas 1” and “pas 2”

First, use summary (yes, you are allowed to!) to argue that accuracy increases faster with objective evidence for PAS 2 than for PAS 1. First I extracted the coefficients from the summary of model 7 and converted them to probabilities using invlogit(). The probability-table states that going from “pas 1” (our baseline/reference point) to “pas 2” increases the accuracy with 36 %. When considering both the effect of target.frames and pas (the interaction), the accuracy when going from “pas 1” to “pas 2” with objective evidence (target.frames:pas2) increases by 61%. Thus, we can conclude that accuracy increases faster with objective evidence for “pas 2” than for “pas 1”. Furthermore, a plot of the predicted probabilities of correct (accuracy), shows that accuracy increases faster with objective evidence for “pas 2”(blue) than “pas 1”(red).

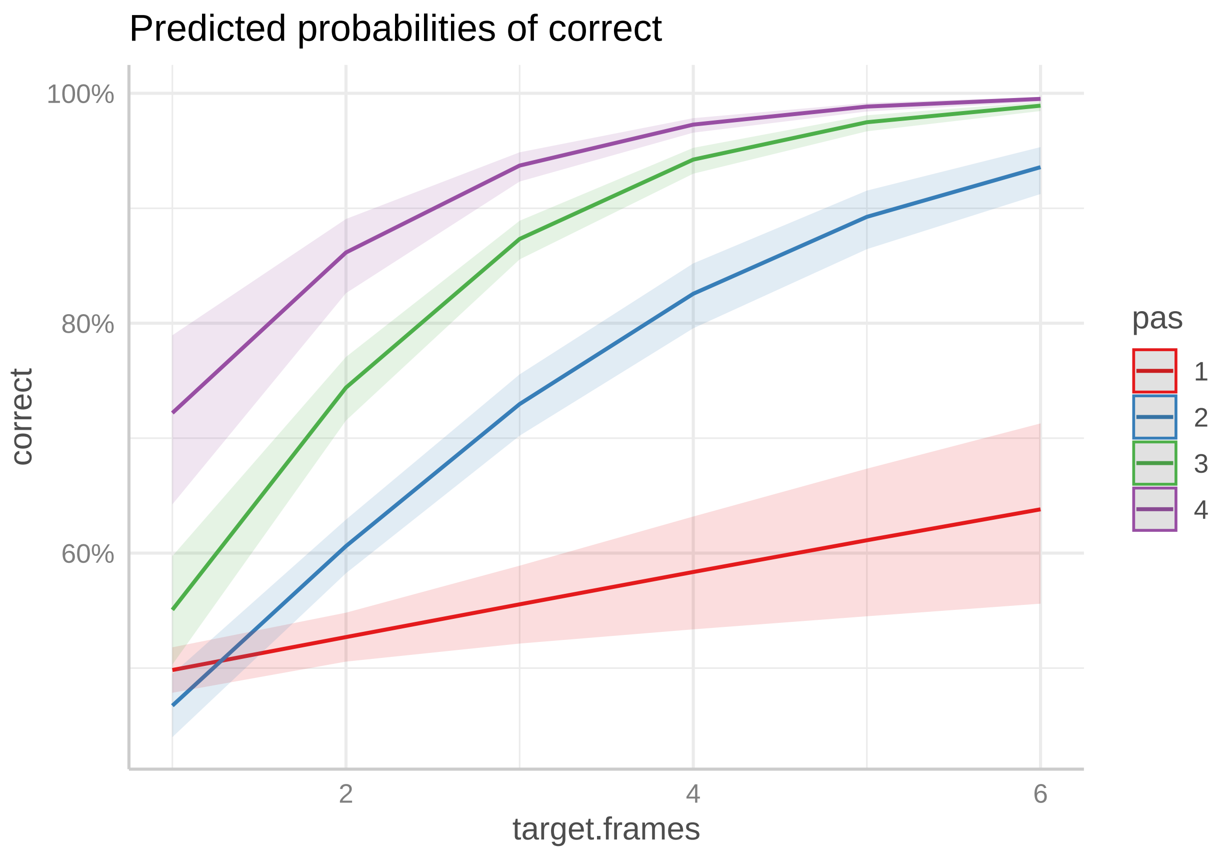
```r
# Fit a model based on the following formula: `correct ~ pas
# * target.frames + (target.frames | subject))`
model7 <- glmer(correct ~ pas * target.frames + (target.frames |
    subject), data = df, binomial(link = "logit"))

summary(model7)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ pas * target.frames + (target.frames | subject)
##    Data: df
##
##      AIC      BIC   logLik deviance df.resid
##  19506.1  19595.5  -9742.0  19484.1    25033
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -19.0111   0.0537   0.1606   0.4849   1.4465
##
## Random effects:
##  Groups  Name          Variance Std.Dev. Corr
##  subject (Intercept)   0.03698  0.1923
##          target.frames 0.02058  0.1434   -0.76
## Number of obs: 25044, groups:  subject, 29
##
## Fixed effects:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.12163    0.06413  -1.897 0.057872 .
## pas2                -0.57139    0.08940  -6.391 1.65e-10 ***
## pas3                -0.53850    0.13959  -3.858 0.000114 ***
## pas4                 0.20157    0.25016   0.806 0.420377
## target.frames        0.11480    0.03708   3.096 0.001960 **
## pas2:target.frames   0.44718    0.03474  12.873  < 2e-16 ***
## pas3:target.frames   0.74869    0.04595  16.293  < 2e-16 ***
## pas4:target.frames   0.75928    0.06840  11.101  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##              (Intr) pas2   pas3   pas4   trgt.f ps2:t. ps3:t.
## pas2         -0.461
## pas3        -0.307  0.247
## pas4        -0.174  0.121  0.091
## target.frms -0.811  0.305  0.207  0.123
## ps2:trgt.fr  0.481 -0.874 -0.244 -0.124 -0.428
## ps3:trgt.fr  0.392 -0.278 -0.891 -0.111 -0.358  0.370
## ps4:trgt.fr  0.276 -0.163 -0.121 -0.918 -0.260  0.225  0.200
```

```
# creating a table showing the percentile increase in
# accuracy for every fixed effect and interactions
estimates <- c(coef(summary(model7))[1:8])
increase_in_prob <- c(invlogit(estimates))
estimates_text <- c("intercept", "pas2", "pas3", "pas4", "target.frames",
    "target.frames:pas2", "target.frames:pas3", "target.frames:pas4")
probability_table <- as_tibble(cbind(estimates_text, increase_in_prob))
print(probability_table)  #comment on these :))
```

```
## # A tibble: 8 x 2
##   estimates_text     increase_in_prob
##   <chr>              <chr>
## 1 intercept          0.469628732818319
## 2 pas2               0.360915790228789
## 3 pas3               0.368537504189747
## 4 pas4               0.550222478099505
## 5 target.frames      0.528668092581941
## 6 target.frames:pas2 0.609969463177935
## 7 target.frames:pas3 0.678893455945173
## 8 target.frames:pas4 0.681197360340063
```

```
# plot of interaction
theme_set(theme_sjplot())
plot_model(model7, type = "pred", terms = c("target.frames",
    "pas"))
```



Exercise 6.2

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. Redoing the test

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. Interpreting accuracy for "pas 2" compared to "pas 3"

Now test the hypothesis that accuracy increases faster with objective evidence for PAS 3 than for PAS 2. (Hypothesis: accuracy changes faster for "target.frames:pas3" than for "target.frames:pas2" with baseline "pas 1") When testing the above hypothesis, a p-value <.05 shows the accuracy does increase faster with objective evidence for "pas 3" than for "pas 2".

iii. Interpreting accuracy for "pas 3" compared to "pas 4"

Also test the hypothesis that accuracy increases faster with objective evidence for PAS 4 than for PAS 3 (Hypothesis: accuracy changes faster for "target.frames:pas4" than for "tar-

get.frames:pas3" with baseline "pas 1") When testing the above hypothesis, a p-value >.05 shows the accuracy does not increase faster with objective evidence for "pas 4" than for "pas 3".

```
# testing if accuracy does increase faster with objective
# evidence for 'pas 2' than for 'pas 1'
contrast.vector1 <- matrix(c(0, 0, 0, 0, 0, 1, 0, 0), nrow = 1)
gh1 <- glht(model7, contrast.vector1)
print(summary(gh1))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: glmer(formula = correct ~ pas * target.frames + (target.frames |
##     subject), data = df, family = binomial(link = "logit"))
##
## Linear Hypotheses:
##        Estimate Std. Error z value Pr(>|z|)
## 1 == 0  0.44718    0.03474   12.87   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

```
# testing if accuracy does increase faster with objective
# evidence for 'pas 3' than for 'pas 2'
contrast.vector2 <- matrix(c(0, 0, 0, 0, 0, -1, 1, 0), nrow = 1)
gh2 <- glht(model7, contrast.vector2)
print(summary(gh2))  #increases sigfinicantly faster
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: glmer(formula = correct ~ pas * target.frames + (target.frames |
##     subject), data = df, family = binomial(link = "logit"))
##
## Linear Hypotheses:
##        Estimate Std. Error z value Pr(>|z|)
## 1 == 0  0.30151    0.04624   6.521 6.98e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

```
# testing if accuracy does increase faster with objective
# evidence for 'pas 4' than for 'pas 3'
contrast.vector3 <- matrix(c(0, 0, 0, 0, 0, 0, -1, 1), nrow = 1)
gh3 <- glht(model7, contrast.vector3)
print(summary(gh3))  # does not increase significantly faster
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: glmer(formula = correct ~ pas * target.frames + (target.frames |
##     subject), data = df, family = binomial(link = "logit"))
```

```
##
## Linear Hypotheses:
##        Estimate Std. Error z value Pr(>|z|)
## 1 == 0  0.01059    0.07438   0.142    0.887
## (Adjusted p values reported -- single-step method)
```

Exercise 6.3

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) The difference between PAS 2 and 1 (tested in 6.1.i) is 0.6099695, which is greater than the difference between PAS 4 and 3 (tested in 6.2.iii) which is 0.5026524. Our test of the hypothesis that accuracy increases faster with objective evidence for PAS 4 than for PAS 3 was also not significant.

```
## testing whether PAS 2 is different from PAS 1
contrast.vector4 <- matrix(c(0, 0, 0, 0, 0, 1, 0, 0), nrow = 1)
gh4 <- glht(model7, contrast.vector4)
print(summary(gh4))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: glmer(formula = correct ~ pas * target.frames + (target.frames |
##     subject), data = df, family = binomial(link = "logit"))
##
## Linear Hypotheses:
##        Estimate Std. Error z value Pr(>|z|)
## 1 == 0  0.44718    0.03474   12.87   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

```
invlogit(coef(summary(gh4))[1])  # probability of increase in accuracy from pas 1 to pas 2 is 60.99695
```

```
##         1
## 0.6099695
```

```
## testing whether PAS 4 is different from PAS 3
contrast.vector5 <- matrix(c(0, 0, 0, 0, 0, 0, -1, 1), nrow = 1)
gh5 <- glht(model7, contrast.vector5)
print(summary(gh5))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: glmer(formula = correct ~ pas * target.frames + (target.frames |
##     subject), data = df, family = binomial(link = "logit"))
##
## Linear Hypotheses:
##        Estimate Std. Error z value Pr(>|z|)
## 1 == 0  0.01059    0.07438   0.142    0.887
## (Adjusted p values reported -- single-step method)
```

```
invlogit(coef(summary(gh5))[1])  # probability of  increase in accuracy from pas 3 to pas 4 is 66. 5118
```

```
##         1
## 0.5026471
```

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).

Exercise 7.1

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 para
‘par‘: you can set _c_ and _d_ as 1. Find good choices for _a_ and _b_ yourself (and argue why they are
‘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
‘upper‘: upper bounds for the four parameters, (the highest value they can take) can set _c_ and _d_ as
```

i. Argumentation for chosen values

The parameters(*par*) are set to be $a = 0.5$ and $b = 1$, since the minimum would 50% chance of getting correct when having enough observations, and the maximum would be 100% chance of getting correct, because it could potentially be possible, hard, but possible. The function (*fn*) to minimize, is the function of a residual sum of squares (RSS). When choosing the *lower bounds* for the four parameters (the lowest value they can take), a and b were set to “0.5”.When choosing the *upper bounds* for the four parameters (the highest value they can take), a and b were set to “1”.

When running the optim-function on the chosen values, the generated estimated parameters are: a = 0.5000, b = 0.9707, c = 2.5089, and d = 0.4310. In conclusion, the optim-function yields an estimate of the lowest accuracy (a) of 50% and the estimated highest accuracy (b) of 97%.

ii. Plotting the fits for “pas” ratings - subject 7

Plot the fits for the PAS ratings on a single plot (for subject 7) `xlim=c(0, 8)`: See chunk

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 `xlim=c(0, 8)` Due to only having 1 subject (subject 7), it is not possible to run the model from exercise 6.1 with random effects (target.frames|subject). Thus, the random effects are removed from the model, in order to run it for subject 7; (correct~pas*target.frames).

See chunk

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

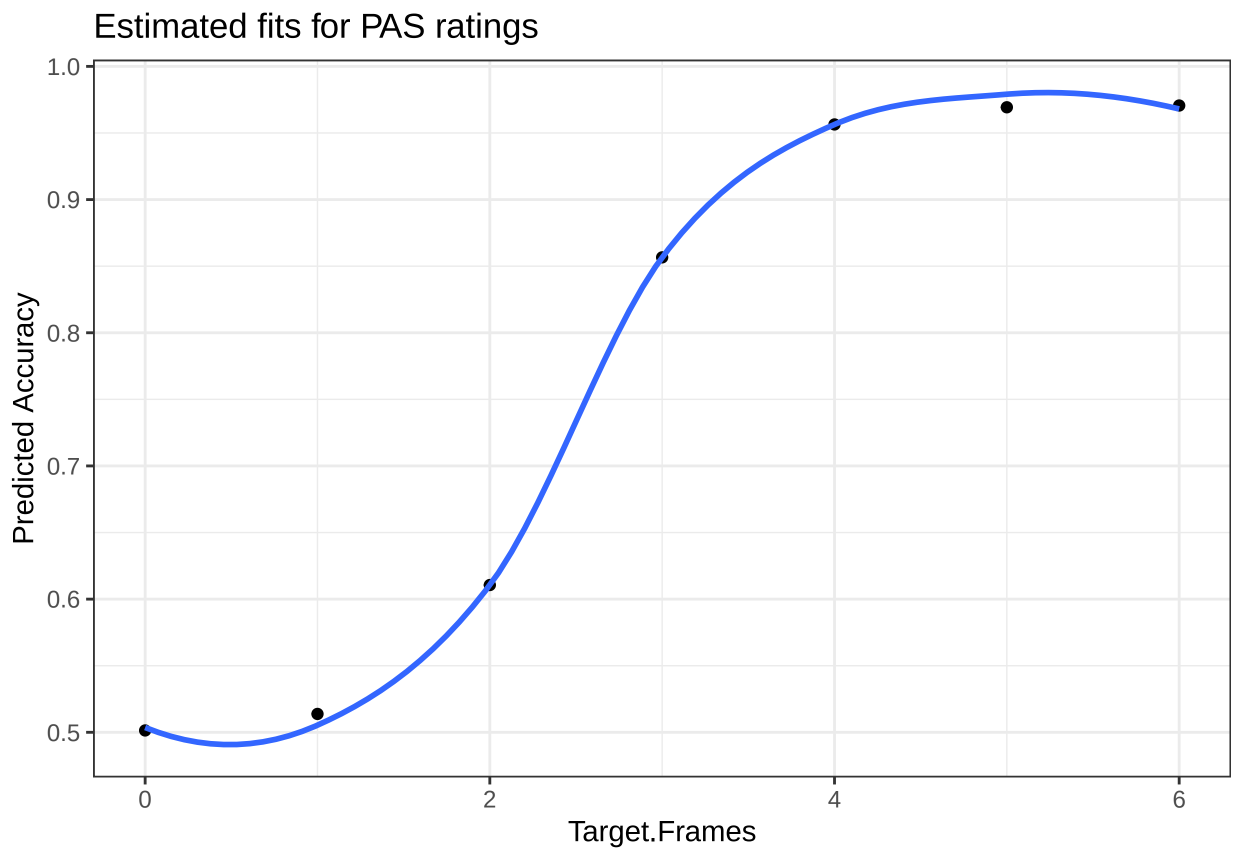
I have absolutely no clue..

```r
set.seed(1)

### function to minimize - function of residual sum of squares
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((c - x)/d)))  #sigmoid funtion
    RSS <- sum((y - y.hat)^2)
    return(RSS)
}


# fit the sigmoid for the four PAS ratings for Subject 7 -
# extract subject 7 data
df7 <- df %>%
    filter(subject == "007") %>%
    dplyr::select(correct, target.frames, pas) %>%
    rename(x = target.frames, y = correct)

df7$y <- as.numeric(df7$y)

# use the function `optim`, to make a list containing the
# four estimated parameters. You should set the following
# arguments:
par <- c(0.5, 1, 1, 1)  # par=  a, b, c, d (starting point)
lower <- c(0.5, 0.5, -Inf, -Inf)  #lower bounds for the four parameters (the lowest value they can take
upper <- c(1, 1, Inf, Inf)  #upper bounds for the four parameters (the highest value they can take), a

parameters <- optim(data = df7, par = par, fn = RSS, method = "L-BFGS-B",
    lower = lower, upper = upper)
print(parameters[[1]][1:4])
```

```
## [1] 0.5000000 0.9707586 2.5089690 0.4310459
```

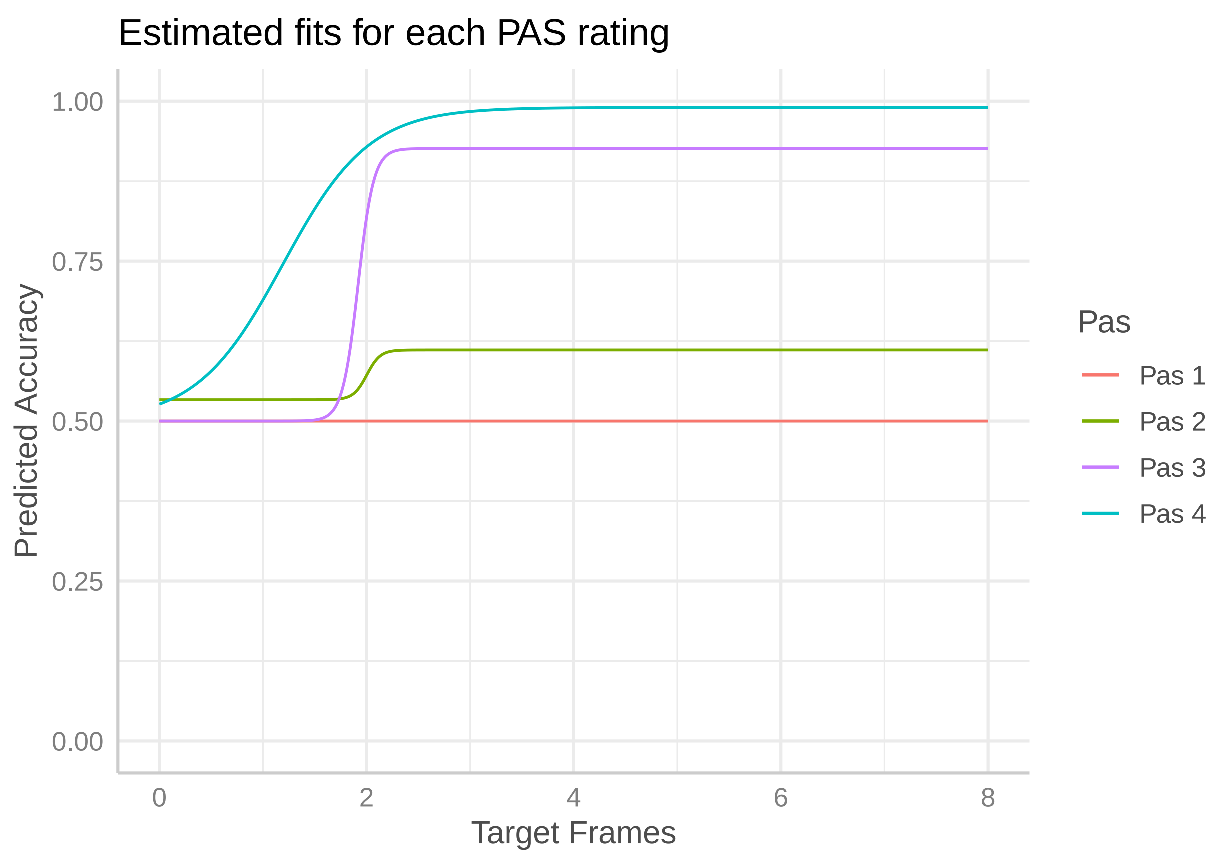
```
########### Estimating fits from sigmoid function

# Estimating the four parameters and making a tibble
parameters <- optim(data = df7, par = par, fn = RSS, method = "L-BFGS-B",
    lower = lower, upper = upper)
parameters_values <- c(parameters[[1]][1], parameters[[1]][2],
    parameters[[1]][3], parameters[[1]][4])

sigmoid_function <- function(x) {
    parameters_values[1] + ((parameters_values[2] - parameters_values[1])/(1 +
        exp((parameters_values[3] - x)/parameters_values[4])))
}

sigplot <- ggplot() + geom_point(aes(x = c(0:6), y = sigmoid_function(0:6))) +
    geom_smooth(aes(x = c(0:6), y = sigmoid_function(0:6)), se = FALSE) +
    labs(title = "Estimated fits for PAS ratings", x = "Target.Frames",
        y = "Predicted Accuracy") + theme_bw()
print(sigplot)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

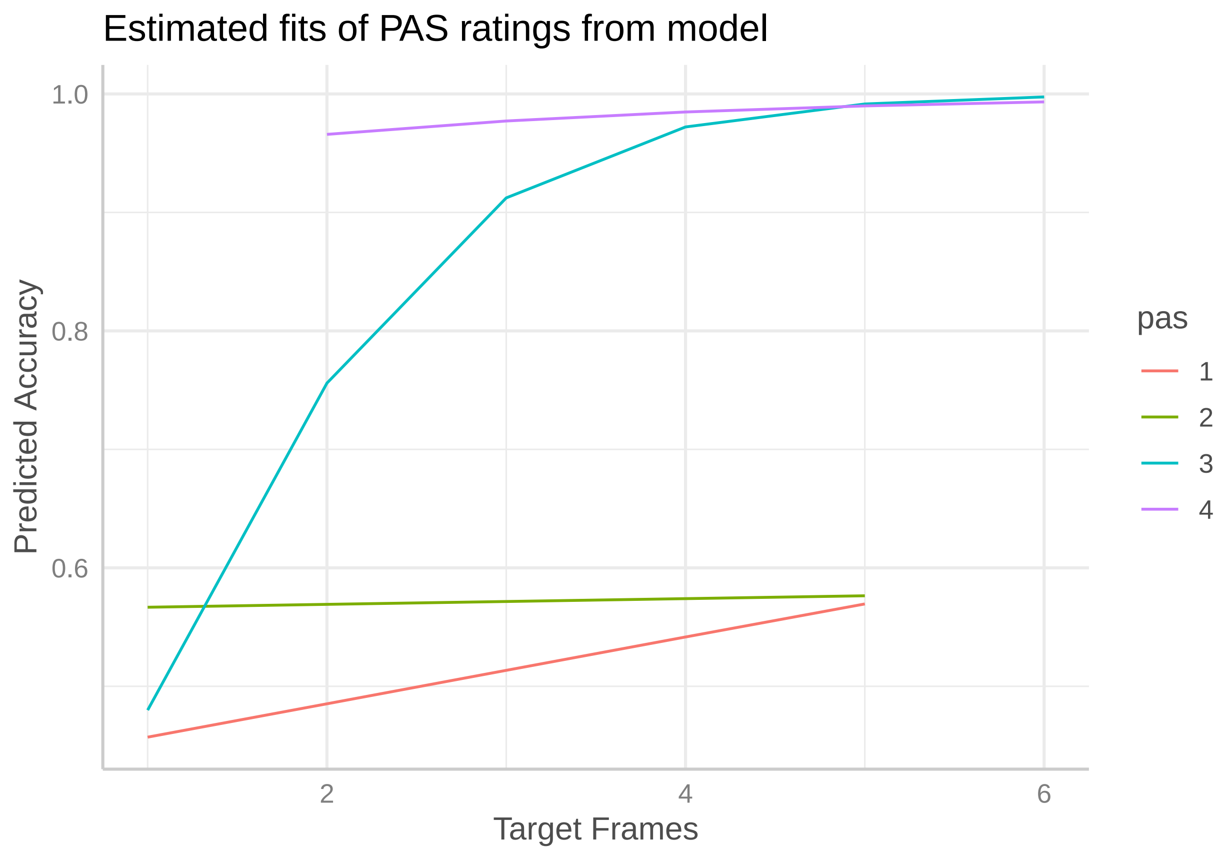


```
########## Plot the fits for the PAS ratings on a single plot (for
########## subject 7) running the optim function on each of the four
########## pas ratings subject 7 - pas 1
pas1 <- df %>%
    filter(subject == "007") %>%
    filter(pas == "1") %>%
    dplyr::select(correct, target.frames, pas) %>%
    rename(x = target.frames, y = correct)

par1 <- optim(data = pas1, par = par, fn = RSS, method = "L-BFGS-B",
    lower = lower, upper = upper)


# subject 7 - pas 2
pas2 <- df %>%
    filter(subject == "007") %>%
    filter(pas == "2") %>%
    dplyr::select(correct, target.frames, pas) %>%
    rename(x = target.frames, y = correct)

par2 <- optim(data = pas2, par = par, fn = RSS, method = "L-BFGS-B",
    lower = lower, upper = upper)


# subject 7 - pas 3
pas3 <- df %>%
    filter(subject == "007") %>%
    filter(pas == "3") %>%
    dplyr::select(correct, target.frames, pas) %>%
    rename(x = target.frames, y = correct)

par3 <- optim(data = pas3, par = par, fn = RSS, method = "L-BFGS-B",
    lower = lower, upper = upper)


# subject 7 - pas 4
pas4 <- df %>%
    filter(subject == "007") %>%
    filter(pas == "4") %>%
    dplyr::select(correct, target.frames, pas) %>%
    rename(x = target.frames, y = correct)

par4 <- optim(data = pas4, par = par, fn = RSS, method = "L-BFGS-B",
    lower = lower, upper = upper)


# creating a function for the sigmoid function
sigmoid_fun <- function(a, b, c, d, x) {
    y.hat <- a + ((b - a)/(1 + exp((c - x)/d)))
    return(y.hat)
}
# simulating x-values between 0-8 with increments of 0.01 for
# making that plot pretty
```

```r
x_values <- seq(0, 8, by = 0.01)

# calculating yhat values for each of the four pas ratings
# using the sigmoid function
yhat1 <- sigmoid_fun(par1[[1]][1], par1[[1]][2], par1[[1]][3],
    par1[[1]][4], x_values)
yhat2 <- sigmoid_fun(par2[[1]][1], par2[[1]][2], par2[[1]][3],
    par2[[1]][4], x_values)
yhat3 <- sigmoid_fun(par3[[1]][1], par3[[1]][2], par3[[1]][3],
    par3[[1]][4], x_values)
yhat4 <- sigmoid_fun(par4[[1]][1], par4[[1]][2], par4[[1]][3],
    par4[[1]][4], x_values)

# combining it all into a dataframe
values <- data.frame(x_values, yhat1, yhat2, yhat3, yhat4)


# plotting the four lines for each of the pas ratings
sigfit_pas_plot <- ggplot(values, aes(x = x_values)) + geom_line(aes(y = yhat1,
    color = "blue")) + geom_line(aes(y = yhat2, color = "green")) +
    geom_line(aes(y = yhat3, color = "red")) + geom_line(aes(y = yhat4,
    color = "orange")) + scale_color_discrete(name = "Pas", breaks = c("blue",
    "green", "red", "orange"), labels = c("Pas 1", "Pas 2", "Pas 3",
    "Pas 4 ")) + xlim(c(0, 8)) + ylim(c(0, 1)) + xlab("Target Frames") +
    ylab("Predicted Accuracy") + ggtitle("Estimated fits for each PAS rating")
print(sigfit_pas_plot)
```

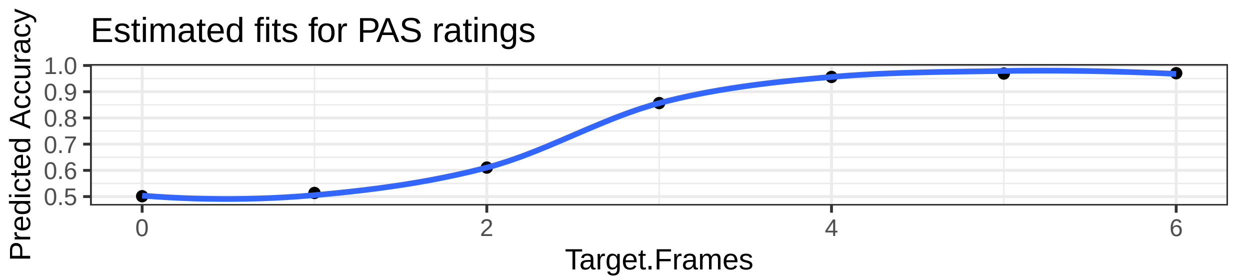


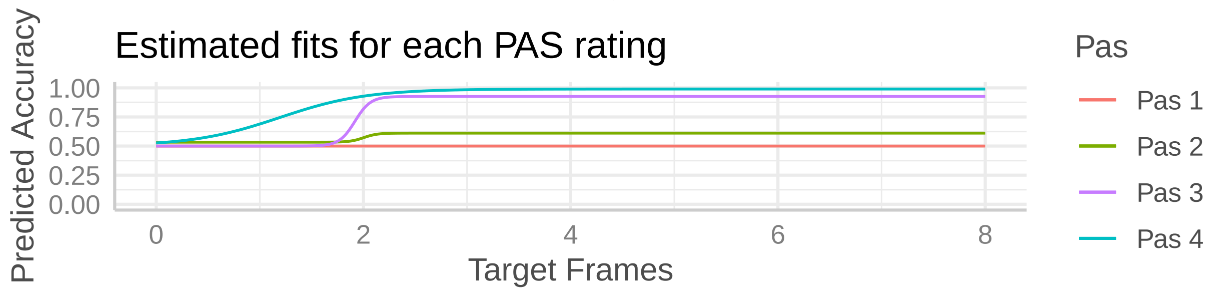
```
########### Create a similar plot for the PAS ratings on a single plot
########### (for subject 7), but this time based on the model from 6.1
########### ‘xlim=c(0, 8)‘

##### Model that gives us predicted values
model8 <- glm(y ~ x * pas, data = df7, binomial(link = "logit"))  #model without random slope and inter
model8_fit <- fitted.values(model8)

###### df for comparison (with our original data)
df7 <- df %>%
    filter(subject == "007") %>%
    dplyr::select(correct, target.frames, pas) %>%
    rename(x = target.frames, y = correct)


# plot of PAS ratings based on model 8 (but only for subject
# 7)
modelfit_pas_plot <- df7 %>%
    ggplot(aes(x = x, y = y, colour = pas)) + geom_line(aes(x = x,
    y = model8_fit)) + labs(title = "Estimated fits of PAS ratings from model",
    x = "Target Frames", y = "Predicted Accuracy")
print(modelfit_pas_plot)
```

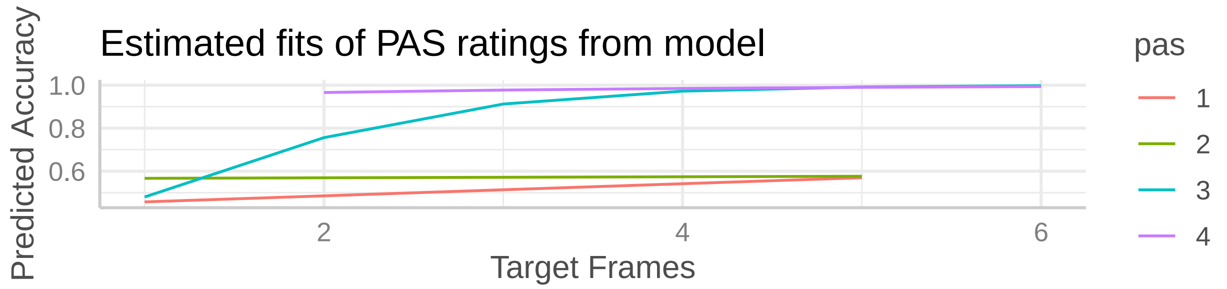


```
# comparing the plots
gridExtra::grid.arrange(sigplot, sigfit_pas_plot, modelfit_pas_plot)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```







Exercise 7.2

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. Comparison and comments

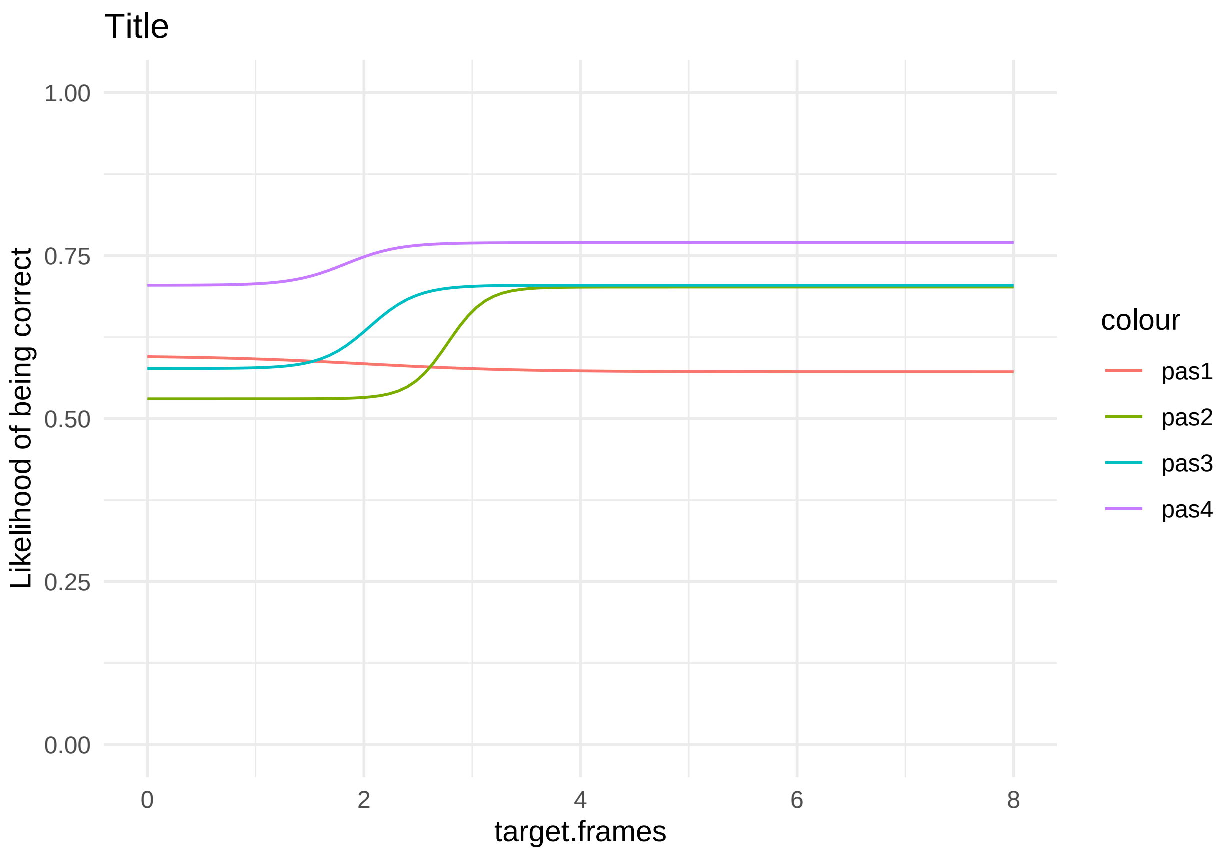
Compare with the figure you made in 5.3.ii and comment on the differences between the fits - mention some advantages and disadvantages of both. Once again, I have absolutely no clue..

```
par <- c(0.5, 1, 1, 1)
fn <- RSS
method = "L-BFGS-B"
lower = c(0.5, 0.5, -Inf, -Inf)
upper = c(1, 1, Inf, Inf)


loop.df <- df %>%
    mutate(x = target.frames, y = correct, subject = as.numeric(subject),
        pas = as.numeric(pas))
n <- 0
```

```
output <- data.frame(subject = character(), pas = integer(),
    a = integer(), b = integer(), c = integer(), d = integer())
for (i in 1:29) {

    for (n in 1:4) {
        subject.df <- loop.df %>%
            filter(subject == i & pas == n)

        optimated <- optim(par = par, data = subject.df, fn = fn,
            method = method, lower = lower, upper = upper)

        optimated.output <- data.frame(subject = i, pas = n,
            a = optimated$par[1], b = optimated$par[2], c = optimated$par[3],
            d = optimated$par[4])

        output <- rbind(output, optimated.output)
    }
}
summarised.output <- output %>%
    group_by(pas) %>%
    summarise(mean.a = mean(a), mean.b = mean(b), mean.c = mean(c),
        mean.d = mean(d))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
# The formula for the sigmoid, with the optimized parameters
# that we found before
mean.fit.pas1 <- function(x) summarised.output$mean.a[1] + ((summarised.output$mean.b[1] -
    summarised.output$mean.a[1])/(1 + exp((summarised.output$mean.c[1] -
    x)/(summarised.output$mean.d[1]))))
mean.fit.pas2 <- function(x) summarised.output$mean.a[2] + ((summarised.output$mean.b[2] -
    summarised.output$mean.a[2])/(2 + exp((summarised.output$mean.c[2] -
    x)/(summarised.output$mean.d[2]))))
mean.fit.pas3 <- function(x) summarised.output$mean.a[3] + ((summarised.output$mean.b[3] -
    summarised.output$mean.a[3])/(3 + exp((summarised.output$mean.c[3] -
    x)/(summarised.output$mean.d[3]))))
mean.fit.pas4 <- function(x) summarised.output$mean.a[4] + ((summarised.output$mean.b[4] -
    summarised.output$mean.a[4])/(4 + exp((summarised.output$mean.c[4] -
    x)/(summarised.output$mean.d[4]))))
ggplot() + xlim(0, 8) + ylim(0, 1) + geom_function(aes(color = "pas1"),
    fun = mean.fit.pas1) + geom_function(aes(color = "pas2"),
    fun = mean.fit.pas2) + geom_function(aes(color = "pas3"),
    fun = mean.fit.pas3) + geom_function(aes(color = "pas4"),
    fun = mean.fit.pas4) + labs(x = "target.frames", y = "Likelihood of being correct",
    title = "Title") + theme_minimal()
```


Title
Likelihood of being correct
1.00
0.75
0.50
0.25
0.00
0
2
4
6
8
target.frames
colour
pas1
pas2
pas3
pas4