Portfolio 2.2, Methods 3, 2021, autumn semester

Laura W. Paaby, studygroup 9

13/10 - 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 (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/ (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 (https://doi.org/10.1016/j.concog.2019.03.007)

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

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)

since we are using read_bulk the missing values are already noted as NA:

```
sum(is.na(data$seed))
```

```
## [1] 2646
```

```
i. Factorise the variables that need factorising
```

```
#checking out the data:
ls.str(data)
```

```
## cue :  num [1:25602] 0 0 0 0 0 0 0 0 0 0 ...
## even.digit :  num [1:25602] 8 4 2 8 6 6 6 4 4 8 ...
## File :  chr [1:25602] "001.csv" "001.csv" "001.csv" "001.csv" "001.csv" "001.csv"
...
## jitter.x :  num [1:25602] -0.431 -0.292 0.308 0.238 -0.427 ...
## jitter.y :  num [1:25602] -0.335 -0.182 0.406 0.175 -0.221 ...
## obj.resp :  chr [1:25602] "e" "o" "e" "o" "e" "o" "o" "e" "e" "o" "e" "o" "e" "o"
"e" ...
## odd.digit :  num [1:25602] 9 5 5 9 3 9 3 5 7 9 ...
## pas :  num [1:25602] 4 4 3 2 3 2 1 1 1 4 ...
## rt.obj :  num [1:25602] 1.418 0.86 0.746 1.676 0.847 ...
## rt.subj :  num [1:25602] 4.93 8.67 8.41 2.13 1.82 ...
## seed :  num [1:25602] 93764 93764 93764 93764 93764 ...
## subject :  chr [1:25602] "001" "001" "001" "001" "001" "001" "001" "001" "001" "00
1" ...
## target.contrast :  num [1:25602] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 ...
## target.frames :  num [1:25602] 9 8 7 6 5 4 3 2 1 9 ...
## target.type :  chr [1:25602] "even" "odd" "even" "odd" "even" "odd" "even" "odd"
"even" ...
## task :  chr [1:25602] "quadruplet" "quadruplet" "quadruplet" "quadruplet" ...
## trial :  num [1:25602] 0 1 2 3 4 5 6 7 8 9 ...
## trial.type :  chr [1:25602] "practice" "practice" "practice" "practice" "practice"
...
```

```
data$trial.type <- as.factor(data$trial.type)
data$pas <- as.factor(data$pas)
data$trial <- as.factor(data$trial)
data$cue <- as.factor(data$cue)
data$task <- as.factor(data$task)
data$target.type <- as.factor(data$target.type)
data$obj.resp <- as.factor(data$obj.resp)
data$subject <- as.factor(data$subject)
```

The factorisation of variables have been made on the basis of Assignment 2 Part 1, in which arguments were presented for the factorisation as well.

```
ii. Remove the practice trials from the dataset (see the _trial.type_ variable)
```

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

```
iii. Create a _correct_ variable
```

```
data_exp$correct <- ifelse((data_exp$obj.resp == "o" & data_exp$target.type =="odd")
 | (data_exp$obj.resp == "e" & data_exp$target.type =="even"), 1,0)
```

```
iv. Describe how the _target.contrast_ and _target.frames_ variables differ compared
to the data from part 1 of this assignment
```

target contrast is now only one value: 0.1. Which is the contrast on the screen between the target and the background. This is now a constant, opposite the previous experiment.

```
summary(data_exp$target.contrast)
```

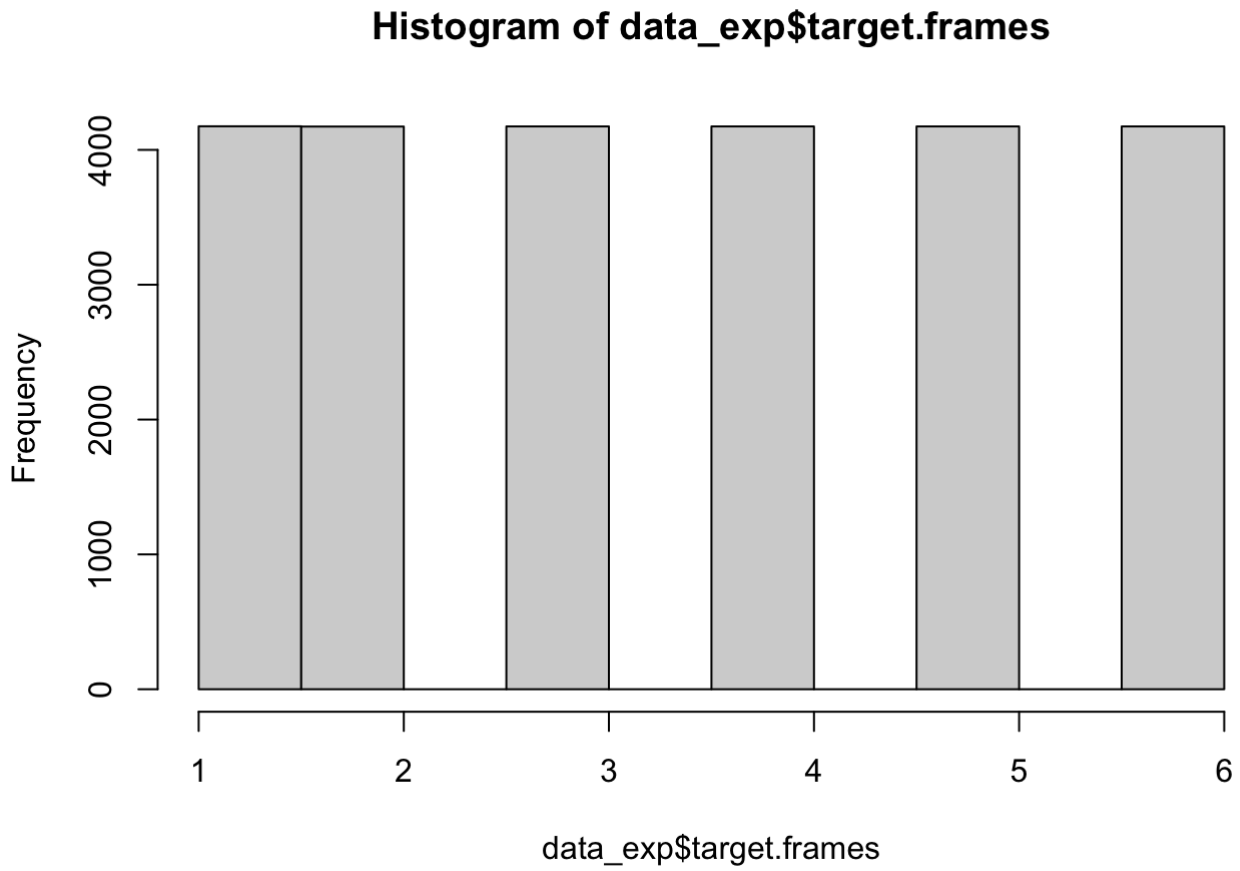
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.1     0.1     0.1     0.1     0.1     0.1
```

target frame are now a ordinal, non-continuous variable, as seen in the summary and the histogram, which also indicates that there is the same amount of counts in each target frame. This is opposite the old dataset, where the frame was a constant. It is the amound of frames in which the target is shown.

```
summary(data_exp$target.frames)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     2.0     3.5     3.5     5.0     6.0
```

```
hist(data_exp$target.frames)
```



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 (**m5.1.pook**) and a partial-pooling model. The partial-pooling (**m5.1.partpool**) model should include a subject-specific intercept.

```
m5.1.pool <- glm(correct ~ target.frames, family = binomial, data = data_exp)
m5.1.partpool <- glmer(correct ~ target.frames + (1 | subject), family = binomial, da
ta = data_exp)
```

```
i. 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.

```
# Likelihood-Function for logistic regression based on the prob-mass-func.
like_funct <- function(model, df, y){
  prob <- fitted(model)

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

```
ii. the log-likelihood-function for logistic regression is: $l(p) = {\displaystyle\su
m_{i=1}^N}[y_i\ln{p}+(1-y_i)\ln{(1-p)}$. Create a function that calculates the log-li
kelihood
```

```
# Log-Likelihood-Function for logistic regression
log_like_funct <- function(model, df, y){
  prob <- fitted(model)

  sum((y*(log(prob)) + (1-y)*log(1-prob)))

}
```

```
iii. apply both functions to the pooling model you just created. Make sure that the l
og-likelihood matches what is returned from the _logLik_ function for the pooled mode
l.
```

```
#using the functions on the pooled model:
pool_prob_like <- like_funct(m5.1.pool, data_exp, data_exp$correct)
pool_prob_likelog <- log_like_funct(m5.1.pool, data_exp, data_exp$correct)

#the outcome:
pool_prob_like
```

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

```
pool_prob_likelog
```

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

Does the likelihood-function return a value that is surprising? *It seems a bit odd that the likelihood estimate is a 0, however considering the huge size of the data and how small the values are, it isn't that suprising after all that it ends on a 0 (also considering the limited precision of the likelihood function).*

```
log_like_r <- logLik(m5.1.pool)

compare_prob <- tibble("Estimated Likelihood Log" =c(pool_prob_likelog), "logLik" = c
(log_like_r))
compare_prob
```

Estimated Likelihood Log	logLik
<dbl>	<dbl>
-10865.25	-10865.25

1 row

Why is the log-likelihood preferable when working with computers with limited precision? *When calculating the likelihood numbers are multiplied together - are these small the numbers will be to small in the end for the computer to comprehend it. Additionally a 0 in this line of numbers will result in it all becoming 0, and not interpretable. Therefore the log-likelihood is preferred, since we then avoid super small numbers and 0's. It simply does not require as much computational precision.*

```
iv. now show that the log-likelihood is a little off when applied to the partial pool
ing model - (the likelihood function is different for the multilevel function - see s
ection 2.1 of https://www.researchgate.net/profile/Douglas-Bates/publication/2753537_
Computational_Methods_for_Multilevel_Modelling/links/00b4953b4108d73427000000/Computa
tional-Methods-for-Multilevel-Modelling.pdf if you are interested)
```

```
log_like_r1 <- logLik(m5.1.partpool)
part_pool1 <- log_like_funct(m5.1.partpool, data_exp, data_exp$correct)

compare_prob1 <- tibble("Estimated Likelihood Log Part Pool" =c(part_pool1), "logLik
 Part Pool" = c(log_like_r1))
compare_prob1
```

Estimated Likelihood Log Part Pool	logLik Part Pool
<dbl>	<dbl>
-10565.53	-10622.03

1 row

As showed, there is a slight difference between the two log-likelihoods

2. Use log-likelihood ratio tests to argue for the addition of predictor variables, start from the null model, `glm(correct ~ 1, 'binomial', data)` (*m2.1*), then **add** subject-level intercepts (*m2.2*), then add a group-level effect of *target.frames* (*m2.3*) and finally add subject-level slopes for *target.frames*.(*m2.4*)

Also assess whether or not a correlation between the subject-level slopes and the subject-level intercepts should be included. (*m2.5*)

```
#the null model
m2.1 <- glm(correct ~ 1, family = binomial, data = data_exp)
llrt2.1 <- log_like_funct(m2.1, data_exp, data_exp$correct)

#added sub level intercepts
m2.2 <- glmer(correct ~ 1 + (1|subject), family = binomial, data = data_exp)
llrt2.2 <- log_like_funct(m2.2, data_exp, data_exp$correct)

#added group-level effect of target frames
m2.3 <- glmer(correct ~ target.frames + (1|subject), family = binomial, data = data_e
xp)
llrt2.3 <- log_like_funct(m2.3, data_exp, data_exp$correct)

#without correlation between slope and intercept (2 times |) and with subject-level e
ffect of target.frames
m2.4 <- glmer(correct ~ target.frames + (1 + target.frames || subject), family = bino
mial, data = data_exp)
llrt2.4 <- log_like_funct(m2.4, data_exp, data_exp$correct)

#equal to the previous, but with correlation between slope and intercept (only one |)
m2.5 <- glmer(correct ~ target.frames + (1 + target.frames | subject), family = binom
ial, data = data_exp)
llrt2.5 <- log_like_funct(m2.5, data_exp, data_exp$correct)

#anova comparison
anov <- anova(m2.2, m2.3, m2.4, m2.5, m2.1)
anov
```

	n... <dbl>	AIC <dbl>	BIC <dbl>	logLik <dbl>	deviance <dbl>	Chisq <dbl>	Df <dbl>	Pr(>Chisq) <dbl>
m2.1	1	26685.08	26693.21	-13341.54	26683.08	*NA*	*NA*	*NA*
m2.2	2	26319.10	26335.35	-13157.55	26315.10	367.97976	1	5.153061e-82
m2.3	3	21250.06	21274.45	-10622.03	21244.06	5071.03518	1	0.000000e+00
m2.4	4	20928.58	20961.09	-10460.29	20920.58	323.48672	1	2.520170e-72
m2.5	5	20907.65	20948.29	-10448.83	20897.65	22.92598	1	1.683609e-06

5 rows

```
#comparison by log likelihood and anova summary
compare_prob2 <- tibble("Model" = c("Null Model m2.1", "Sub Intercept Model m2.2", "T
arget Frames Effect m2.3", "No Correlation between Random Effects m2.4", "Correlation
Between Random Effects m2.5"), "Log Likelihood RT" =c(llrt2.1, llrt2.2, llrt2.3, llrt
2.4, llrt2.5))
compare_prob2
```

Model <chr>	Log Likelihood RT <dbl>
Null Model m2.1	-13341.54
Sub Intercept Model m2.2	-13105.03

Model <chr>	Log Likelihood RT <dbl>
Target Frames Effect m2.3	-10565.53
No Correlation between Random Effects m2.4	-10378.96
Correlation Between Random Effects m2.5	-10375.14

5 rows

```
i. write a short methods section and a results section where you indicate which model you chose and the statistics relevant for that choice.
```

When comparing models by their log-likelihood ratio, one should look for the higher value, since one wants it to be maximized. In this case I would therefore suggest model 2.5: glmer(correct ~ target.frames + (1 + target frames|subject), family = binomial) which has the target frames as the fixed effects, subject as the random intercept and a correlation between the subject level slopes and intercepts: which has a log likelihood ratio value of -10375.14.

Additionally … it can be seen how the results of an anova comparison indicates the same finding - the model perform best in the anova as well, leading me to suggest that a correlation should be included in our model

Summed upe: **The model with the highest log-likelihood score (-10375.14) was found to be the mixed effect model which included a correlation between random slope and intercept. Due to this finding, the m2.4 model is chosen as the final model with: beta_0 = 1.09 (SE = 0.059, p < .001) and beta_1 = 0.83 (SE = 0.044, p < .001).**

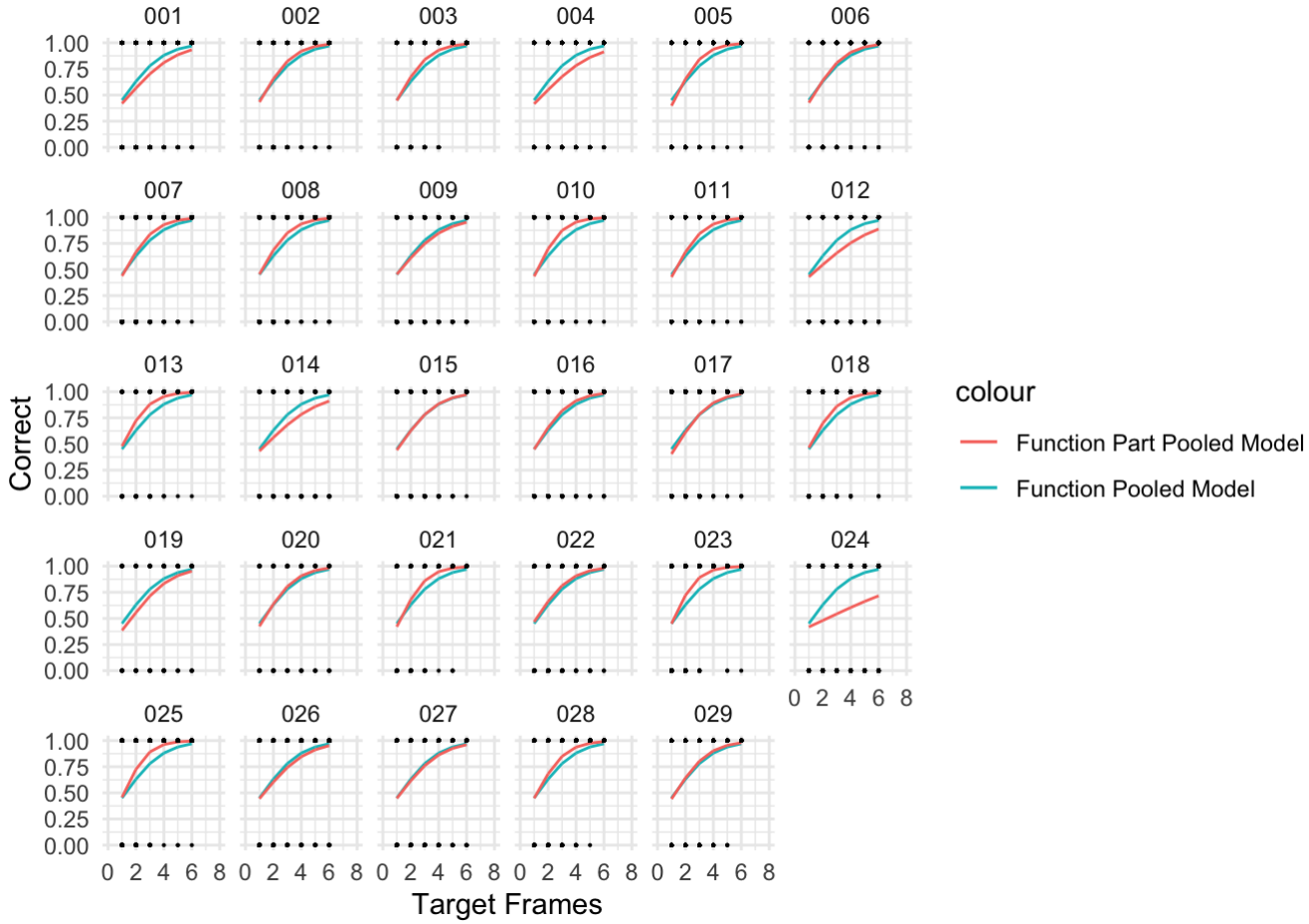
as found in:

```
summary(m2.4)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ target.frames + (1 + target.frames || subject)
##    Data: data_exp
##
##      AIC      BIC   logLik deviance df.resid
##  20928.6  20961.1 -10460.3  20920.6    25040
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -14.1977   0.0898   0.2470   0.4894   1.3292
##
## Random effects:
##  Groups    Name          Variance Std.Dev.
##  subject   (Intercept)   0.0360   0.1897
##  subject.1 target.frames 0.0366   0.1913
## Number of obs: 25044, groups:  subject, 29
##
## Fixed effects:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.06926    0.05083  -21.04   <2e-16 ***
## target.frames  0.82207    0.03817   21.54   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## target.frms -0.230
```

Include a plot of the estimated group-level function with `xlim=c(0, 8)` that includes the estimated subject-specific functions.

```
#fitted values from the pool model to compare
pool.fit <- fitted(m5.1.pool)

#fitted values over the chosen model
m5.fit <- fitted(m2.5)


plot2 <- ggplot(data_exp, aes(target.frames, as.numeric(as.character(correct)))) +
    geom_line(aes(x = target.frames, y = pool.fit, color = "Function Pooled Model"))+
    geom_line(aes(x = target.frames, y = m5.fit, color = "Function Part Pooled Model"
))+
    geom_point(aes(y = as.numeric(as.character(correct), color = "Response")), size =
0.07) +
   facet_wrap(~subject)+
  xlim(min = 0, max = 8)+
    theme_minimal()+
    xlab("Target Frames")+
    ylab('Correct')


plot2
```



The dots: Are the responses given by each participant (0 being wrong, 1 correct)

By eyeballing the plot it appears as if the function for subject 24 looks rather different that the group-specific function. One could additionally compare the subject specific function with the group specific, and find that in many cases there is a bit of a difference.

```
ii. also include in the results section whether the fit didn't look good for any of t
he subjects. If so, identify those subjects in the report, and judge (no statistical
test) whether their performance (accuracy) differed from that of the other subjects.
Was their performance better than chance? (Use a statistical test this time) (50 %)
```

```
data_24 <- data_exp %>%
  filter(subject == "024")

t.test(x = data_24$correct, mu = 0.5)
```

```
##
##  One Sample t-test
##
## data:  data_24$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
```

Inspecting the one-sampled t-test, we see whether the performance of subject 24 was better than if the outcomes were due to pure chance (hence the 0.5, which is the value it should be compared by) - this indicates there is a significant difference ($p < .001$) between the subject 24 performance and a performance achieved by chance.

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

```
#pas as group level effect
m3.1 <- glmer(correct ~ target.frames + pas + (1 + target.frames | subject), family =
binomial, data = data_exp)

#pas as interaction with target.frames
m3.2 <- glmer(correct ~ target.frames * pas + (1 + target.frames | subject), family =
binomial, data = data_exp)
```

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

it appears to be working :D

```
#does log-like justify????
log_compare <- tibble("Model" = c("m2.5", "m.3.1", "m3.2"), "Type" = c("Correlation M
odel without Pas", "With Pas as Group Level Effect", "With Pas As Interaction"), "Log
-Like Value" = c(log_like_funct(m2.5, data_exp, data_exp$correct), log_like_funct(m3.
1, data_exp, data_exp$correct), log_like_funct(m3.2, data_exp, data_exp$correct)))

log_compare
```

Model <chr>	**Type** <chr>	**Log-Like Value** <dbl>
m2.5	Correlation Model without Pas	-10375.136
m.3.1	With Pas as Group Level Effect	-9860.225
m3.2	With Pas As Interaction	-9684.128

3 rows

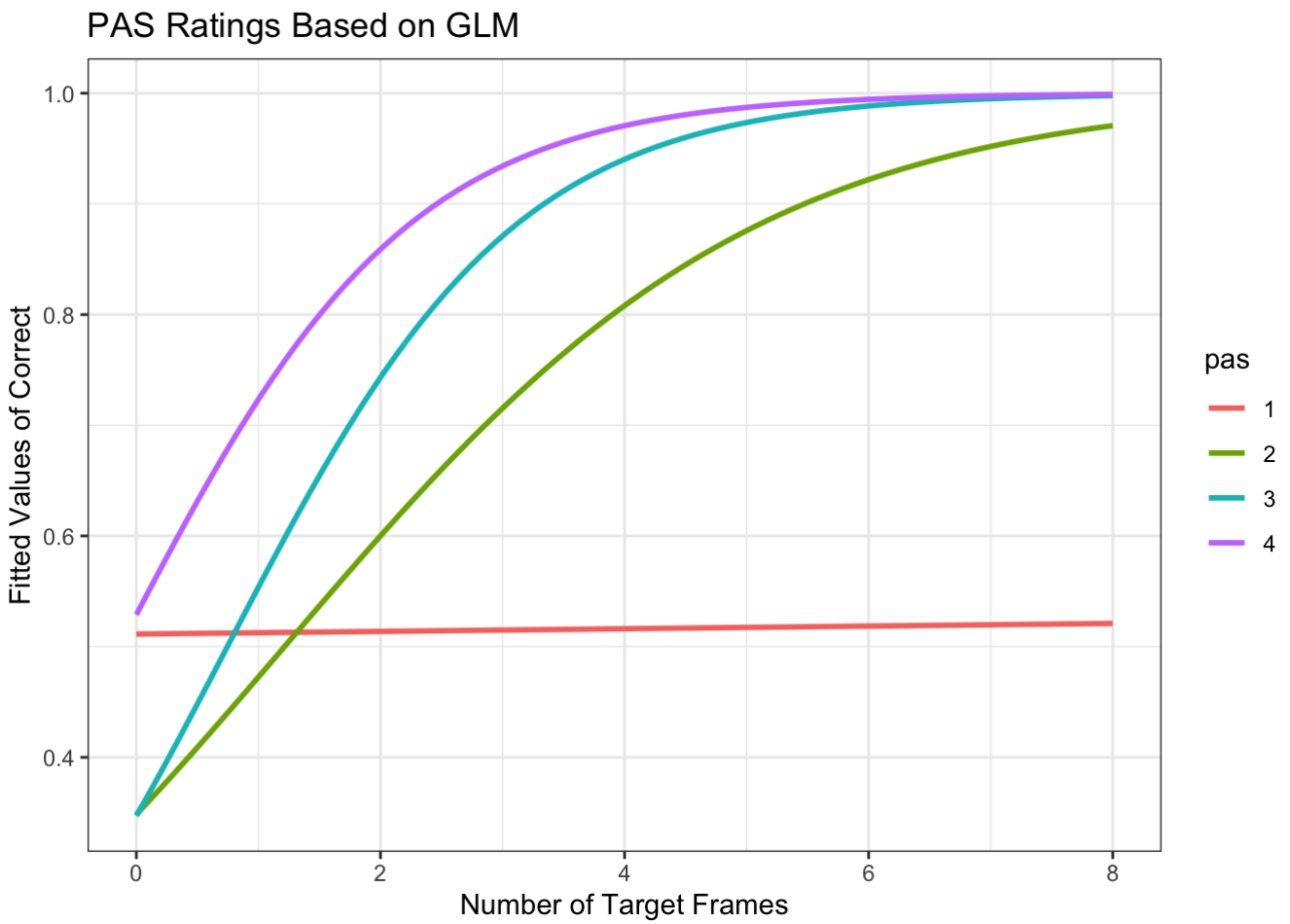
*Seeing how the Log-Likelihood Value increases by adding PAS as interaction, it can be justified to add this to our model. Our final model is now m3.2: glmer(correct ~ target.frames * pas + (1 + target.frames | subject), family = binomial).*

```
ii. 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.
```

```
data_exp$fit_mod <- fitted.values(m3.2)

plot_5.3 <- data_exp %>%
  ggplot(aes(target.frames, fit_mod)) +
  geom_smooth(aes(colour = pas), method = "glm", method.args = list(family = "binomia
l"), se = FALSE, fullrange = TRUE) +
  xlim(0,8) +
  ylab("Fitted Values of Correct") +
  xlab("Number of Target Frames") +
  ggtitle("PAS Ratings Based on GLM")+
  theme_bw()

plot_5.3
```

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

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```



This visualize how subjects performs better the higher the PAS value. Additionally the amount of target.frames appears to affect the performance: the more target frames, the better the performance - apart from when pas = 1, where it seems as if most subjects have given the wrong answer. However this is ONLY by eyeballing the

plot. The chosen model m3.2 includes an interaction between pas and target.frames, and their individual effect. Looking at the plot, this choice of model is supported: at pas 1 target frames barely affects correctness, while it in the other pas affects it a lot.

Also comment on the estimated functions' behaviour at target.frame = 0 - is that behaviour reasonable?
since the lowest amount of targetframes in the experiment is 0, Id argue that it would make sense to look at the behaviour to that targetframe. Should we inspect it however, we see how it appears to have values under due to chance, which again is not reasonable. Unless you are subject 24 :). We should additionally remember that these values are estimated and not observed in the actual experiment.

```
#summary of the chosen model:
summary(m3.2)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ target.frames * pas + (1 + target.frames | subject)
##    Data: data_exp
##
##      AIC      BIC   logLik deviance df.resid
##  19506.1  19595.5  -9742.0  19484.1    25033
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -19.0116   0.0537   0.1607   0.4849   1.4465
##
## Random effects:
##  Groups  Name          Variance Std.Dev. Corr
##  subject (Intercept)   0.03697  0.1923
##          target.frames 0.02057  0.1434   -0.76
## Number of obs: 25044, groups:  subject, 29
##
## Fixed effects:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.12164    0.06419  -1.895 0.058081 .
## target.frames       0.11480    0.03710   3.095 0.001971 **
## pas2               -0.57138    0.08948  -6.386 1.71e-10 ***
## pas3               -0.53844    0.13980  -3.852 0.000117 ***
## pas4                0.20147    0.25132   0.802 0.422758
## target.frames:pas2  0.44718    0.03477  12.863  < 2e-16 ***
## target.frames:pas3  0.74867    0.04602  16.270  < 2e-16 ***
## target.frames:pas4  0.75930    0.06867  11.057  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) trgt.f pas2   pas3   pas4   trg.:2 trg.:3
## target.frms -0.811
## pas2        -0.462  0.306
## pas3        -0.308  0.208  0.249
## pas4        -0.175  0.124  0.123  0.091
## trgt.frms:2  0.482 -0.429 -0.874 -0.246 -0.125
## trgt.frms:3  0.393 -0.359 -0.279 -0.891 -0.111  0.371
## trgt.frms:4  0.276 -0.260 -0.164 -0.121 -0.919  0.226  0.200
```

The log-likelihoods from this can now be used to find the estimated behaviour when target.frame = 0

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

#intercept - so when targetframe equals 0
inv.logit(-0.12164)
```

```
## [1] 0.4696274
```

```
#pas 2. here i can see that the probability of answering correct when going from pas
 1 to pas 2 is increased with 36%
inv.logit(-0.57138)
```

```
## [1] 0.3609185
```

```
#pas 3: Here i can see that the probability of answering correct when going from pas
 2 to pas 3 is increased with 36%
inv.logit(-0.53844)
```

```
## [1] 0.3685506
```

```
#pas 4:  here i can see that the probability of answering correct when going from pas
1 to pas 2 is increased with 55%
inv.logit(0.20147)
```

```
## [1] 0.5501978
```

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 neighboring 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.

```
#making the model
m6.1 <- glmer(correct ~ pas * target.frames + (1+target.frames | subject), family = b
inomial(link = "logit"), data = data_exp)

summary(m6.1)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ pas * target.frames + (1 + target.frames | subject)
##    Data: data_exp
##
##      AIC      BIC   logLik deviance df.resid
##  19506.1  19595.5  -9742.0  19484.1    25033
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -19.0110   0.0537   0.1606   0.4849   1.4465
##
## Random effects:
##  Groups  Name          Variance Std.Dev. Corr
##  subject (Intercept)   0.03698  0.1923
##          target.frames 0.02057  0.1434   -0.76
## Number of obs: 25044, groups:  subject, 29
##
## Fixed effects:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.12163    0.06416  -1.896 0.058008 .
## pas2               -0.57140    0.08943  -6.389 1.67e-10 ***
## pas3               -0.53848    0.13956  -3.859 0.000114 ***
## pas4                0.20156    0.25059   0.804 0.421216
## target.frames       0.11480    0.03709   3.095 0.001966 **
## pas2:target.frames  0.44719    0.03475  12.869  < 2e-16 ***
## pas3:target.frames  0.74869    0.04595  16.293  < 2e-16 ***
## pas4:target.frames  0.75928    0.06850  11.084  < 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        -0.462
## pas3        -0.308  0.248
## pas4        -0.174  0.121  0.092
## target.frms -0.811  0.306  0.208  0.124
## ps2:trgt.fr  0.482 -0.874 -0.245 -0.124 -0.428
## ps3:trgt.fr  0.393 -0.279 -0.891 -0.112 -0.359  0.371
## ps4:trgt.fr  0.276 -0.163 -0.121 -0.918 -0.260  0.225  0.200
```

For pas = 1, the accuracy increases with 0.1148 on the logit scale per increase in target.frames, this is an increase with a probability of almost 53%. For pas = 2 the increase is 0.1148+0.4472 = 0.562 on the logit scale per increase in target.frames, which is probability is around 53,5%. This indicates how the accuracy increases faster for pas 2 compared to pas 1 per increase in target frames.

```
######from logit to prob
#for pas 1:
est_1<- (coef(summary(m6.1))[5])
est.inv <- inv.logit(est_1)
est.inv
```

```
## [1] 0.5286684
```

```
# for pas 2:
est_2<- (coef(summary(m6.1))[5+6])
est_2_inv <- inv.logit(est_2)
est_2_inv
```

```
## [1] 0.5348322
```

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:

```
summary(m6.1)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ pas * target.frames + (1 + target.frames | subject)
##    Data: data_exp
##
##      AIC      BIC   logLik deviance df.resid
##  19506.1  19595.5  -9742.0  19484.1    25033
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -19.0110   0.0537   0.1606   0.4849   1.4465
##
## Random effects:
##  Groups  Name          Variance Std.Dev. Corr
##  subject (Intercept)   0.03698  0.1923
##          target.frames 0.02057  0.1434   -0.76
## Number of obs: 25044, groups:  subject, 29
##
## Fixed effects:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.12163    0.06416  -1.896 0.058008 .
## pas2               -0.57140    0.08943  -6.389 1.67e-10 ***
## pas3               -0.53848    0.13956  -3.859 0.000114 ***
## pas4                0.20156    0.25059   0.804 0.421216
## target.frames       0.11480    0.03709   3.095 0.001966 **
## pas2:target.frames  0.44719    0.03475  12.869  < 2e-16 ***
## pas3:target.frames  0.74869    0.04595  16.293  < 2e-16 ***
## pas4:target.frames  0.75928    0.06850  11.084  < 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        -0.462
## pas3        -0.308  0.248
## pas4        -0.174  0.121  0.092
## target.frms -0.811  0.306  0.208  0.124
## ps2:trgt.fr  0.482 -0.874 -0.245 -0.124 -0.428
## ps3:trgt.fr  0.393 -0.279 -0.891 -0.112 -0.359  0.371
## ps4:trgt.fr  0.276 -0.163 -0.121 -0.918 -0.260  0.225  0.200
```

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 <- glht(m6.1, contrast.vector)
print(summary(gh))

inv.logit(0.44719)
```

this now give us an outcome who shows how there is a significant difference in how fast the accuracy increases when going from PAS 1 to PAS 2 (p < .001) Where the shift should be made are given by the 1 in the contrast vector. By taking the inverse logit of the estimate we find that the probability of an increase from PAS 1 till 2 to be: 60,997%

```
ii. Now test the hypothesis that accuracy increases faster with objective evidence fo
r PAS 3 than for PAS 2.
```

as another example, we could also test whether there is a difference in the increment of the accuracy between PAS 2 and PAS 3. This is now done by changing the place of the 1's in the contrast vector

```
contrast.vector1 <- matrix(c(0, -1, 1, 0, 0, 0, 0, 0), nrow=1)
gh1 <- glht(m6.1, contrast.vector1)
print(summary(gh1))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: glmer(formula = correct ~ pas * target.frames + (1 + target.frames |
##     subject), data = data_exp, family = binomial(link = "logit"))
##
## Linear Hypotheses:
##        Estimate Std. Error z value Pr(>|z|)
## 1 == 0  0.03292    0.14587   0.226    0.821
## (Adjusted p values reported -- single-step method)
```

```
inv.logit(0.03292)
```

```
## [1] 0.5082293
```

It can here be concluded that there is no significant difference between PAS 2 and 3 (p > .05) By taking the inverse logit of the estimate we find that the probability of an increase from PAS 2 till 3 to be: 50,823%

this can now be checked for the slope as well - before it was only intercept:

```
contrast.vector11 <- matrix(c(0, 0, 0, 0, 0, -1, 1, 0), nrow=1)
gh11 <- glht(m6.1, contrast.vector11)
print(summary(gh11))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: glmer(formula = correct ~ pas * target.frames + (1 + target.frames |
##     subject), data = data_exp, family = binomial(link = "logit"))
##
## Linear Hypotheses:
##        Estimate Std. Error z value Pr(>|z|)
## 1 == 0  0.30150    0.04621   6.524 6.85e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

Here we see how the difference in slope are significant (p < .001)

```
iii. Also test the hypothesis that accuracy increases faster with objective evidence
for PAS 4 than for PAS 3
```

```
#between pas 3 and 4
contrast.vector2 <- matrix(c(0, 0, 0, 0, 0, 0, -1, 1), nrow=1)
gh2 <- glht(m6.1, contrast.vector2)
print(summary(gh2))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: glmer(formula = correct ~ pas * target.frames + (1 + target.frames |
##     subject), data = data_exp, family = binomial(link = "logit"))
##
## Linear Hypotheses:
##        Estimate Std. Error z value Pr(>|z|)
## 1 == 0  0.01060    0.07445   0.142    0.887
## (Adjusted p values reported -- single-step method)
```

```
inv.logit(0.01060)
```

```
## [1] 0.50265
```

This is not significant, indicating that the perception in both pass 3 and 4 is enough for the subject to have the same precision in their accuracy of the answers (correct vs. wrong)

By taking the inverse logit of the estimate we find that the probability of an increase from PAS 3 till 4 to be: 50,26%

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)

```
#taking the log of the gh values to make them comparable:
logit <- function(x) log(x / (1 - x))
log_gh <- inv.logit(0.44719)
log_gh2 <- inv.logit(0.01060)

#so if there is a difference between gh and gh2:
compare_gh <- tibble("Pas" = c("Pas 2 and 1", "Pas 4 and 3"), "GH Value" = c("0.4363"
, 0.002838), "Logged Value" = c(log_gh, log_gh2))
compare_gh
```

Pas	**GH Value**	**Logged Value**
<chr>	<chr>	<dbl>
Pas 2 and 1	0.4363	0.6099709
Pas 4 and 3	0.002838	0.5026500

2 rows

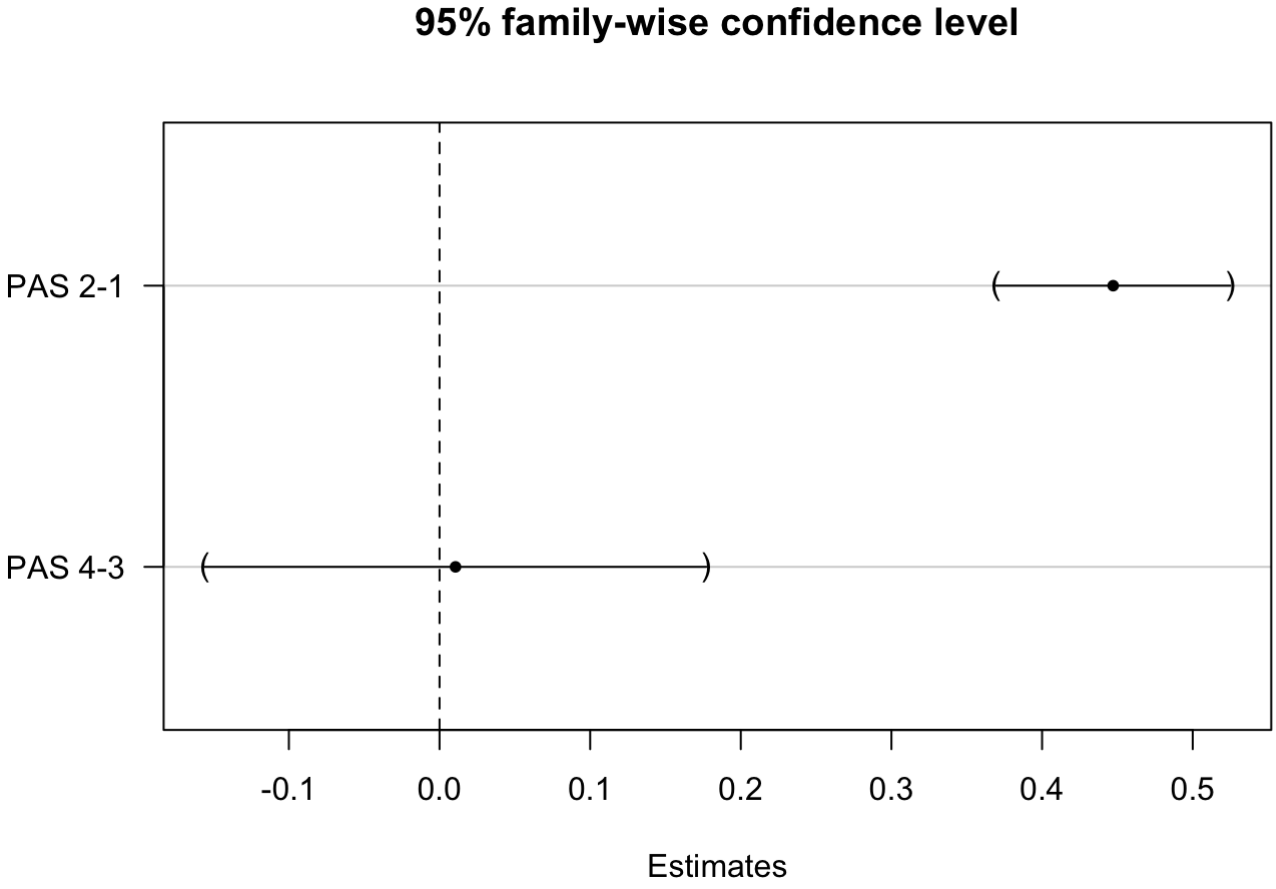
*There is clearly is a difference, but to test if they are significally different more should be done …

```
#binding the two pas matrixes into one
contrast.matrix <- rbind(c(0, 0, 0, 0, 0, 1, 0, 0), c(0, 0, 0, 0, 0, 0, -1, 1))
rownames(contrast.matrix) <- c("PAS 2-1", "PAS 4-3")
gh <- glht(m6.1, contrast.matrix)

print(summary(gh))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: glmer(formula = correct ~ pas * target.frames + (1 + target.frames |
##     subject), data = data_exp, family = binomial(link = "logit"))
##
## Linear Hypotheses:
##              Estimate Std. Error z value Pr(>|z|)
## PAS 2-1 == 0  0.44719    0.03475  12.869   <1e-10 ***
## PAS 4-3 == 0  0.01060    0.07445   0.142    0.987
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

```
#now we see that the estimates clearly are different which can be plottet:
```

```
# a vizualization of the difference:
plot(gh, xlab= "Estimates")
```



Eyeballing the plot we see no overlap in the two 95% confidence intervals, which indicates a 'true' difference between the differences of PAS2-PAS1 (0.44718) and PAS4-PAS3 (0.01060).

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
    ## you fill in the estimate of y.hat
    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

```
#making a new dataframe only for participant 7
data_7.1 <- data_exp %>%
  dplyr::select(subject, pas, target.frames, correct) %>%
  rename(x = target.frames, y = correct)

dataset7 <- data_7.1 %>%
  filter(subject == "007")

### making a new data set for each pas for subject 7

sub_7_pas_1 <- dataset7 %>%
  filter(subject == "007" & pas == 1)

sub_7_pas_2 <- dataset7 %>%
  filter(subject == "007" & pas == 2)

sub_7_pas_3 <- dataset7 %>%
  filter(subject == "007" & pas == 3)

sub_7_pas_4 <- dataset7 %>%
  filter(subject == "007" & pas == 4)
```

i. use the function `optim`. It returns a list that among other things contains the f
our 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 t
hey are appropriate)

```
##
optim1 <- optim(c(0.5, 1.00, 1.00, 1.00), fn = RSS, data = sub_7_pas_1, method = 'L-B
FGS-B', lower = c(0.5, 0.5, -Inf, -Inf), upper = c(1, 1, Inf, Inf))

optim2 <- optim(c(0.5, 1.00, 1.00, 1.00), fn = RSS, data = sub_7_pas_2, method = 'L-B
FGS-B', lower = c(0.5, 0.5, -Inf, -Inf), upper = c(1, 1, Inf, Inf))

optim3 <- optim(c(0.5, 1.00, 1.00, 1.00), fn = RSS, data = sub_7_pas_3, method = 'L-B
FGS-B', lower = c(0.5, 0.5, -Inf, -Inf), upper = c(1, 1, Inf, Inf))

optim4 <- optim(c(0.5, 1.00, 1.00, 1.00), fn = RSS, data = sub_7_pas_4, method = 'L-B
FGS-B', lower = c(0.5, 0.5, -Inf, -Inf), upper = c(1, 1, Inf, Inf))


## printing the optimal values for each:
print(c(optim1, optim2, optim3, optim4))
```

```
## $par
## [1] 0.500000 0.500000 2.218342 1.449114
##
## $value
## [1] 45.75
##
## $counts
## function gradient
##        5        5
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: NORM OF PROJECTED GRADIENT <= PGTOL"
##
## $par
## [1] 0.53333057 0.61111627 2.00265034 0.06488018
##
## $value
## [1] 31.75556
##
## $counts
## function gradient
##       66       66
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
##
## $par
## [1] 0.5000000 0.9259250 1.9182875 0.0754563
##
## $value
## [1] 7.476435
##
## $counts
## function gradient
##       37       37
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
##
## $par
## [1] 0.5000000 0.9900976 1.1927144 0.4163145
##
## $value
## [1] 5.871986
##
## $counts
```

```
## function gradient
##       45       45
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

```
#the upper and lower in here are set to 0 and 1, since it is a sigmoid function.
```

Argument: the minimum accuracy level is set to 0,5 since it is then what would be due to chance (50%). The maximum is set to 1 since I find i reasonable that a few might get an accuracy of 100%, and no one will for sure be above.

This now give us new suggested parameters, that will be the most optimal to use for each pas

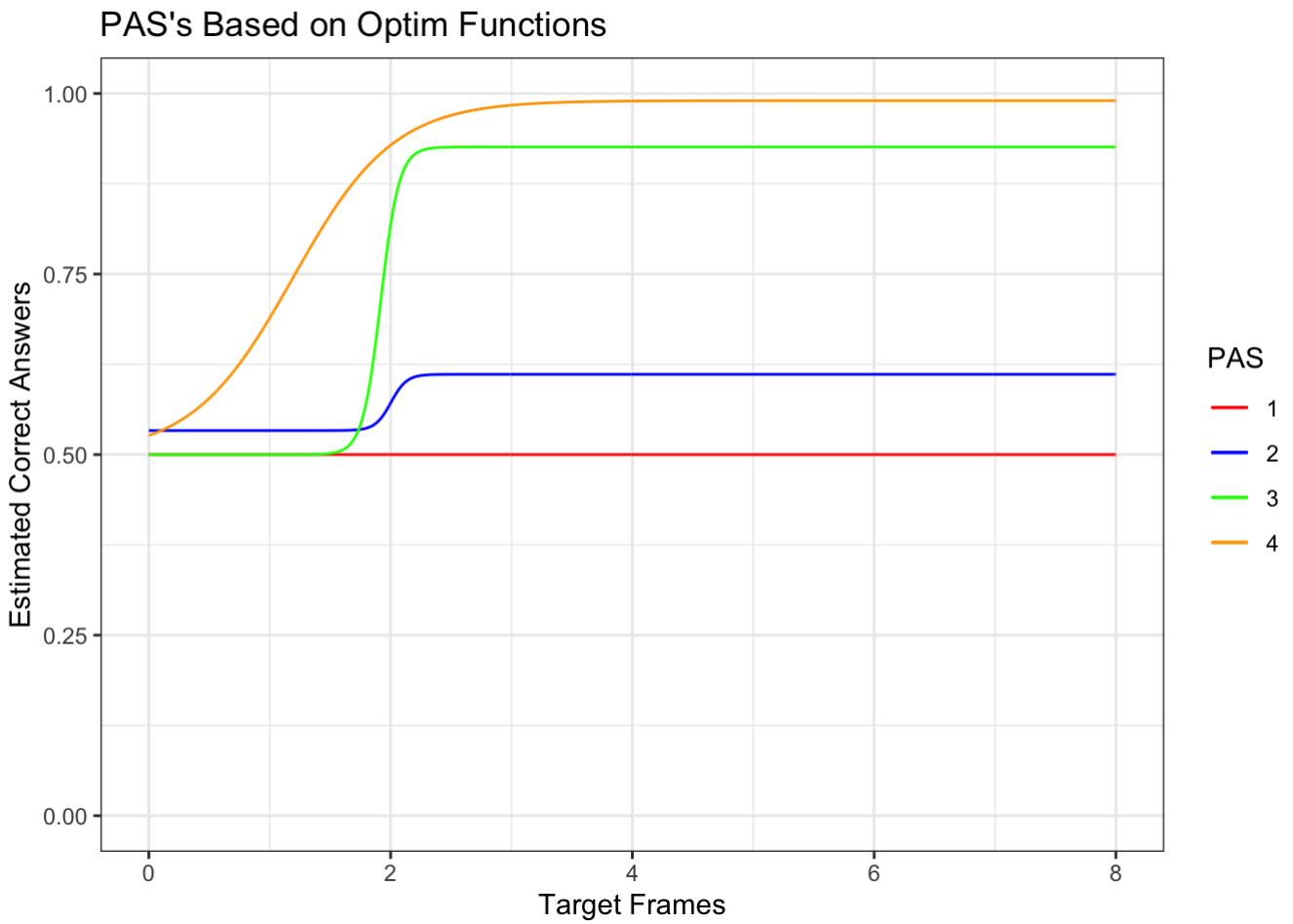
```
ii. Plot the fits for the PAS ratings on a single plot (for subject 7) `xlim=c(0, 8)`
```

to do so we must first calculate the y hats:

```
y_hat_func <- function(a, b, c, d, x) {
  y.hat <- a + ((b-a)/(1+exp(1)^((c-x)/d)))
  return(y.hat)
}

#making and empty frame for the yhats:
optim_data <- data.frame(cbind("x" = seq(0, 8, by = 0.01)))

#for pas1
optim_data$yhat1 <- y_hat_func(optim1$par[1], optim1$par[2], optim1$par[3],optim1$par
[4], optim_data$x)
#for pas2
optim_data$yhat2 <- y_hat_func(optim2$par[1], optim2$par[2], optim2$par[3],optim2$par
[4], optim_data$x)
#for pas3
optim_data$yhat3 <- y_hat_func(optim3$par[1], optim3$par[2], optim3$par[3],optim3$par
[4], optim_data$x)
#for pas4
optim_data$yhat4 <- y_hat_func(optim4$par[1], optim4$par[2], optim4$par[3],optim4$par
[4], optim_data$x)
```

this can now be plottet ….

```
plot_optim <- ggplot(optim_data) +
  geom_line(aes(x=x, y=yhat1, color = "1"))+
  geom_line(aes(x=x, y=yhat2, color = "2"))+
  geom_line(aes(x=x, y=yhat3, color = "3"))+
  geom_line(aes(x=x, y=yhat4, color = "4"))+
  scale_color_manual(name = "PAS", values = c("1" = "red", "2" = "blue", "3" = "gree
n", "4" = "orange"))+
  xlim(c(0,8))+
  ylim(c(0,1))+
  labs(y = "Estimated Correct Answers", x = "Target Frames", title = "PAS's Based on
 Optim Functions")+
  theme_bw()


plot_optim
```

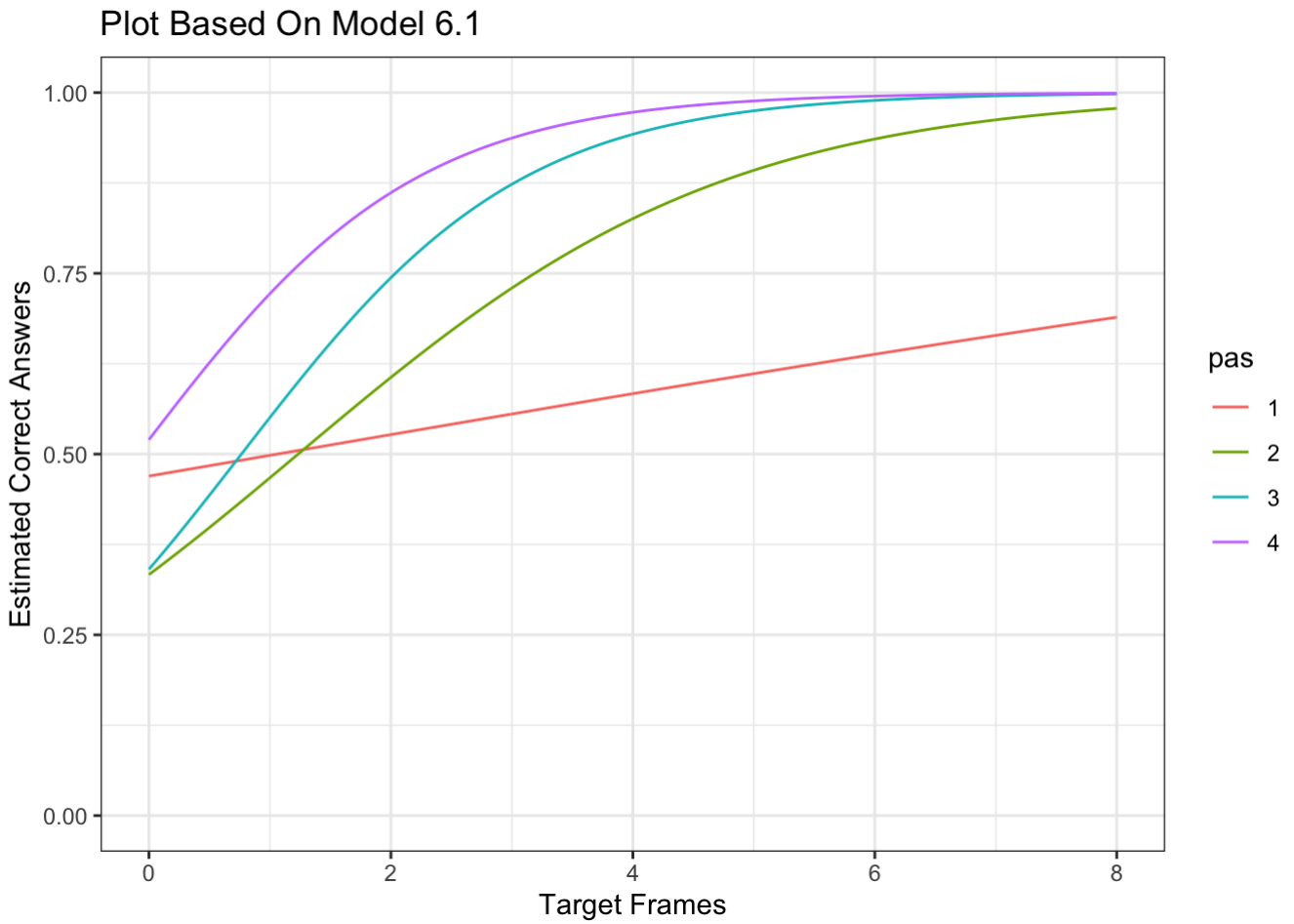



```
iii. Create a similar plot for the PAS ratings on a single plot (for subject 7), but
this time based on the model from 6.1 `xlim=c(0, 8)`
```

to do so (getting an xlim 0-8) we must make predictions for the y values (correct) for each pas per subject 7, across all for pass. I have chosen to use a model fitted on all the data (so all subjects) and then using it to predict new y values, specific for subject 7. This is done to avoid fitting the model to the same data twice and making the prediction within sample - if I fit on subject 7 only and then afterwards try to predict on subject 7 I get the same values. Additionally, subjects are taken into account in the big data the model is fitted on, thus I find this to be the solution.

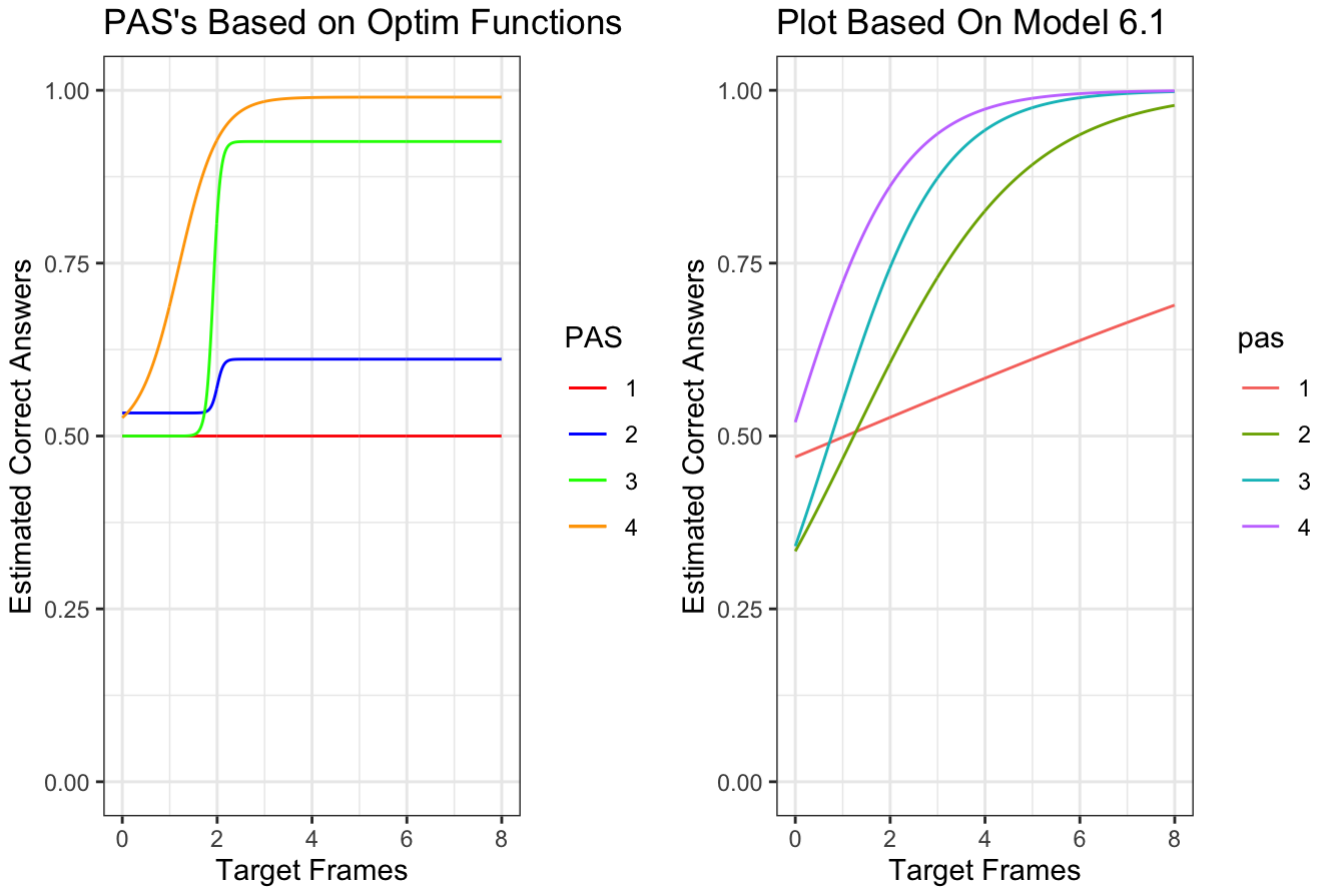
```
#to get all the points we neew to make the lines go from 0-8 on the x axe we generate
the data:
model6_data_new <- data.frame(cbind("x" = seq(0, 8, by = 0.01),"pas1" = rep(1,801),"p
as2" = rep(2,801), "pas3" = rep(3,801), "pas4" = rep(4,801), "subject" = rep(7,801)))

##combining all the passes to make them work with the model:
model6_pivot <- model6_data_new %>%
  pivot_longer(cols = pas1:pas4, values_to = "pas")

## model:
m6.1_new <- glmer(y ~ pas * x + (1+x | subject), family = binomial(link = "logit"), d
ata = data_7.1)

## factorising:
model6_pivot$pas <- as.factor(model6_pivot$pas)
model6_pivot$subject <- as.factor(model6_pivot$subject)


model6_pivot$yhat_model6 <- predict(m6.1_new, re.form = ~(1+x| subject), newdata = mo
del6_pivot, type = "response", allow.new.levels = TRUE)
```

```
#plot
plot6.1 <- model6_pivot %>%
  ggplot(aes(x = x, y = yhat_model6, colour = pas)) +
  xlim(c(0,8))+
  ylim(c(0,1))+
  geom_line(aes(x = x, y = yhat_model6))+
  labs(title = "Plot Based On Model 6.1",
       x = "Target Frames",
       y = "Estimated Correct Answers")+
  theme_bw()

plot6.1
```



This can now be compared to the sigmoid function plot over pas ratings.

```
cool_com <- ggarrange(plot_optim, plot6.1)
annotate_figure(cool_com, top = text_grob("PAS fits for Subject 7", color = "darkblu
e", face = "italic", size = 18))
```



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

The plot appears to have quite the same tendencies, at first glance. However, by a closer look especially pas 2 differs a lot in its relation to the other lines in the two plots. In the optim function the correctness is quite low, whereas it is almost as good as in pas 3 in the model 6.1 plot.

By looking at the Optim plot, we see how the optim function has fixed the problem of having less correct answers then if due to chance, which I'd say is a big advantage! This stems from us setting the accepted minimum at a=0.5. This however can't be done when the correctness of answers are estimated on the general mixed model m6.1, thus we see the slopes being below 0.5 in the model 6.1 plot.

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 (https://doi.org/10.1016/j.concog.2019.03.007))

```
#making a loop that creates pas ratings estimated by the optim function per subject
cool_loop_function <- function(){
  data_frame_parameters <- data.frame(subject = NA, pas = NA , a = NA, b = NA, c = NA
, d = NA)


  for (i in 1:4){
    df_temp1 <- data_exp %>%
      filter(pas == i) %>%
      mutate(subject = as.numeric(subject))

    for (ii in 1:length(unique(data_exp$subject))){
      data_temp <- df_temp1 %>%
        filter(subject == ii) %>%
        dplyr::select(target.frames, correct, pas) %>%
        rename(x = target.frames, y = correct)

      op_temp = optim(par = c(0.5,0.5,1,1), fn = RSS, data = data_temp, method = "L-B
FGS-B", lower = c(0.5,0.5,-Inf,-Inf),
      upper = c(1,1,Inf,Inf))
      data_frame_parameters <- rbind(data_frame_parameters , c(ii,i,op_temp$par))

    }
  }
  return(data_frame_parameters)
}

df_loop_pasratings <- cool_loop_function() %>%
  na.omit() #removing na's



df_loop_pasratings
```

	subject	pas	a	b	c	d
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
2	1	1	0.5000000	0.5000000	1.0000000	1.0000000000
3	2	1	0.6072665	0.5000000	0.9206736	0.7154031594
4	3	1	0.6248181	0.5374096	0.8602455	0.4300550152
5	4	1	0.7593121	0.5000000	0.9151872	0.0932777361
6	5	1	0.5000000	0.5000000	1.0000000	1.0000000000
7	6	1	0.6143188	0.5000000	1.0058600	0.9558662745
8	7	1	0.5000000	0.5000000	1.0000000	1.0000000000
9	8	1	0.6165075	0.5000000	0.9436198	0.8557934710
10	9	1	0.5000000	0.5391376	1.7117410	0.0203760864
11	10	1	0.5000000	0.5000000	1.0000000	1.0000000000

this now gives us a dataframe containing the parameters for each subject in each pas, which we then can take the mean of:

```
#taking the average of all subjects ratings per pas:
mean_rating <- df_loop_pasratings %>%
  group_by(pas) %>%
  summarise(a = mean(a), b = mean(b), c = mean(c), d = mean(d))
```

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

```
mean_rating
```

pas <dbl>	a <dbl>	b <dbl>	c <dbl>	d <dbl>
1	0.5796431	0.5514403	1.638810	0.4571415
2	0.5274088	0.8775388	2.832406	0.1749703
3	0.5768916	0.9556266	2.281409	0.2259202
4	0.7198609	0.9658502	2.190136	0.5012006

4 rows

Having the mean of each parameter in each class we can now estimate the y values for all subjects, which is originally estimated by the optim.

```
#making y values per pass:
y_for_pas <- function() {
  x <- seq(1, 8, 0.1)
  y_df = data.frame(x)

  for (i in 1:4){
  df_temp <- mean_rating %>%
      filter(pas == i)

   y_df[,i+1] <- y_hat_func(df_temp$a, df_temp$b, df_temp$c, df_temp$d, x)

  }
y_df <- y_df %>%
  rename("1" = V2, "2" = V3, "3" = V4, "4" = V5)

return(y_df)

}

df_est_pas <- y_for_pas()
df_est_pas
```

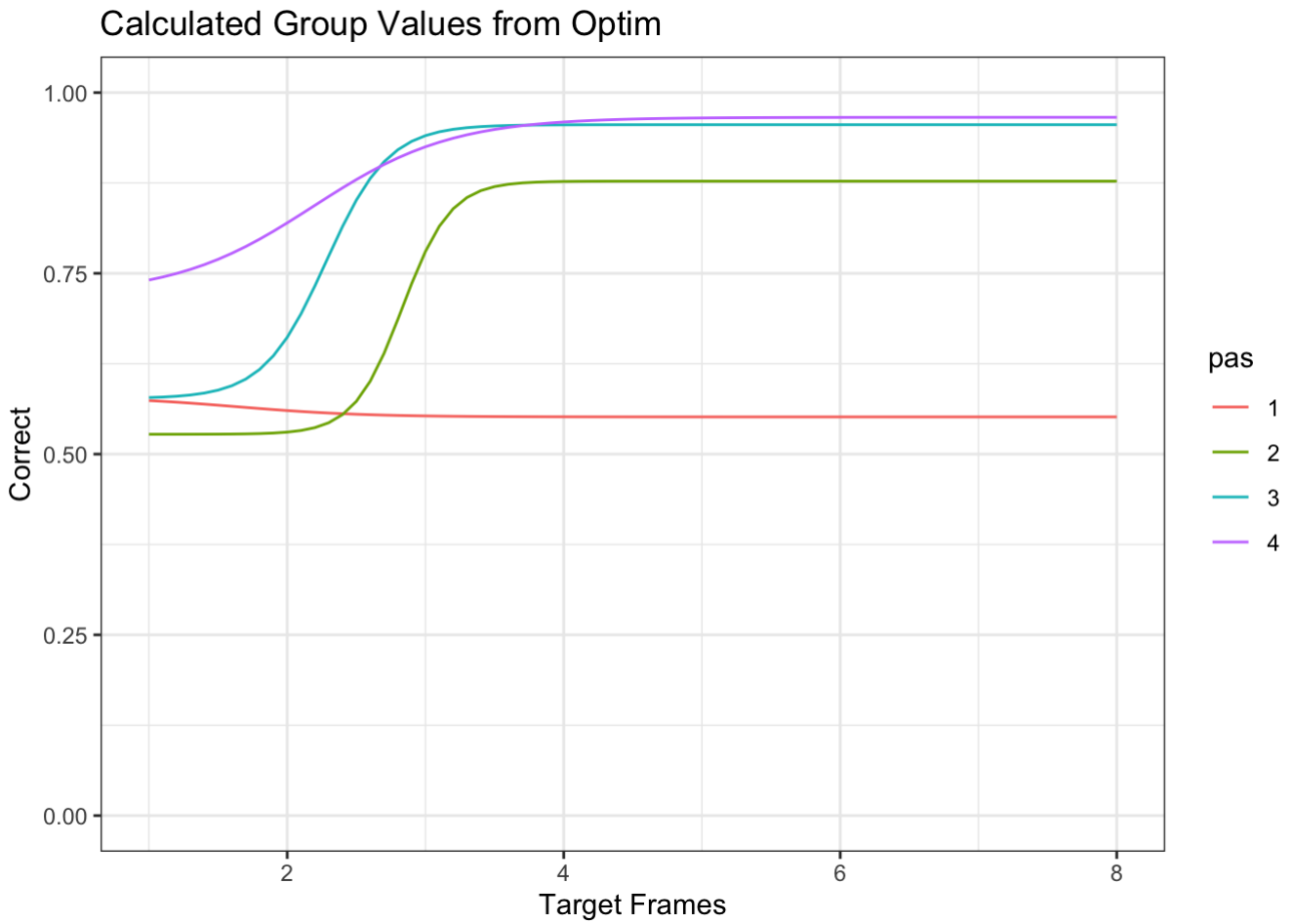
x <dbl>	1 <dbl>	2 <dbl>	3 <dbl>	4 <dbl>
1.0	0.5740525	0.5274187	0.5781904	0.7408026

x <dbl>	1 <dbl>	2 <dbl>	3 <dbl>	4 <dbl>
1.1	0.5730071	0.5274264	0.5789097	0.7449551
1.2	0.5718338	0.5274399	0.5800241	0.7498215
1.3	0.5705406	0.5274638	0.5817460	0.7554794
1.4	0.5691434	0.5275062	0.5843954	0.7619973
1.5	0.5676663	0.5275813	0.5884460	0.7694269
1.6	0.5661399	0.5277142	0.5945791	0.7777943
1.7	0.5645993	0.5279493	0.6037297	0.7870907
1.8	0.5630810	0.5283649	0.6170872	0.7972649
1.9	0.5616194	0.5290985	0.6359767	0.8082177

1-10 of 71 rows Previous **1** 2 3 4 5 6 … 8 Next

Plotting the estimated function at the group-level by taking the mean for each of the four parameters, a, b, c and d across subjects:

```
df_final <- df_est_pas %>%
  pivot_longer(cols = c("1", "2","3", "4") , names_to = "pas", values_to = "y_hat_mer
ged")

plot_final <- df_final %>%
  ggplot(aes(x, y_hat_merged, color = pas))+
  geom_line() +
  ylim(0,1)+
  labs(title = "Calculated Group Values from Optim",
       x = "Target Frames",
       y = "Correct") +
  theme_bw()

plot_final
```



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.

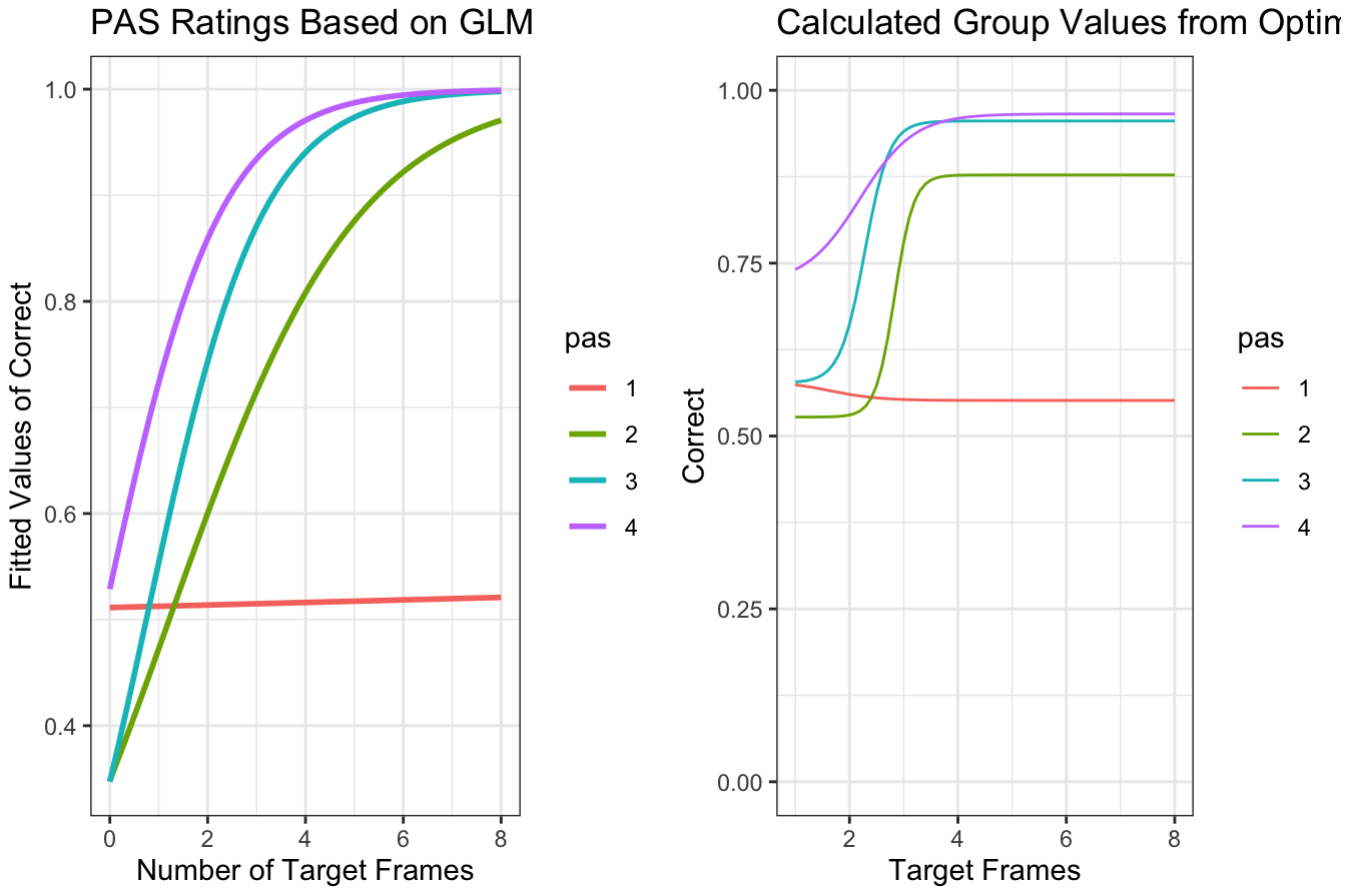
```
last_com <- ggarrange(plot_5.3 , plot_final)
```

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

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
annotate_figure(last_com, top = text_grob("Comparison of PAS Plots", color = "darkblu
e", face = "italic", size = 18))
```

Comparison of PAS Plots



Just by eyeballing the plots, the lines for each pas is quite similar in relationship to one another. Looking at the optim group plot at the right the line for pas 1 is weird looking, since it between target frame 1-3 it declines, since a is higher than b (as found in df_loop_pasratings). However in this plot (group optim) the lines are not at any point below 0.5 on the y axes, which is an advantages the GLM plot does not have - this means that the estimated accuracy of correct answers never gets worse than by due to chance. Moreover, The GLM plot doesn't seem to act as an entire sigmoid function either, which I intuitively would say it should, to reflect the data properly - an ability the group optim plot possesses.