portfolio 2.2, Methods 3, 2021, autumn semester

Sigurd Fyhn Sørensen

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

```
pacman::p_load(tidyverse, lmerTest, lme4, stargazer, carret, readbulk, TinyTex, rstan
arm, reabulk)
```

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)

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 identified some na's in the seed variable.

```
sum(is.na(df_exp1$seed))
```

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

precisely 2646. So no need to change anything as the NA's has aleady been added to the seed variable.

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

```
df_exp1 <- df_exp1 %>%
  mutate(right_answer = if_else(target.type == "odd" & obj.resp == "o" | target.type
 == "even" & obj.resp == "e",1,0)) %>%
  mutate(right_answer = as.numeric(right_answer))


df_exp1 <- df_exp1 %>%
  mutate(right_answer = as.numeric(right_answer)) %>%
  mutate(subject = as.factor(subject)) %>%
  mutate(task = as.factor(task)) %>%
  mutate(target.type = as.factor (target.type)) %>%
  mutate(trial.type = as.factor(trial.type)) %>%
  mutate(pas = as.factor(pas))
```

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

```
levels(df_exp1$trial.type)
```

```
## [1] "experiment" "practice"
```

```
df_exp1 <- df_exp1 %>%
  filter(trial.type == "experiment")
```

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

This has been done in the *r preprocess* chunk.

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

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

In experiment 2 (part 1 of this assignment) **target.frames** = 3 in all trials. In experiment 1 (part 2 of this assignment) **target.frame** = (1:6) across trials.

In experiment 2 (part 1 of this assignment) target.contrast: Min. 1st Qu. Median Mean 3rd Qu. Max. 0.01000 0.05683 0.06329 0.08771 0.09392 1.00000

In experiment 1 (part 2 of this assignment) target.contrast: Min. 1st Qu. Median Mean 3rd Qu. Max. 0.1 0.1 0.1 0.1 0.1 0.1

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.

```
m5.1.1 <- glmer(right_answer ~ target.frames + (1|subject)+ (1|trial), data = df_exp
1, family = binomial(link = "logit"))

m5.1.2 <- glm(right_answer ~ target.frames, data = df_exp1, family = binomial(link =
"logit"))
```

```
i. the likelihood-function for logistic regression is: $L(p)={\displaystyle\prod_{i=
1}^Np^{y_i}(1-p)^{(1-y_i)}}$ (Remember the probability mass function for the Bernoull
i Distribution). Create a function that calculates the likelihood.
```

```
likelihood_function <- function(model, y){
  p = fitted.values(model)
  temp_vec = p^y*(1-p)^(1-y)
  return(prod(temp_vec))
}
```

```
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 <- function(model, y){
  p = fitted.values(model)
  temp_vec = y * log(p) + (1-y) * log(1-p)
  return(sum(temp_vec))
}
```

```
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. Does the likelihood-function return a value that is surprising? Why is the log-lik
elihood preferable when working with computers with limited precision?
```

```
likelihood_function(m5.1.2, df_exp1$right_answer)
```

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

```
log_likelihood_function(m5.1.2, df_exp1$right_answer)
```

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

I wouldn't say it is surprising that the likelihood function returns 0. If one iteration of i in the product chain $L(p) = \prod_{i=1}^{N} p^{y_i}(1-p)^{(1-y_i)}$ would be 0 the final product would result in 0.

The argument for preferring log-likelihood is much similair to the argument of why likelihood returns 0. Taking the log not only simplifies the subsequent mathematical analysis, but it also helps numerically because the product of a large number of small probabilities can easily be lesser than the numerical precision of the computer, and this is resolved by computing instead the sum of the log probabilities.

Log-likelihood is also preferential in regards of machine learning and optimization. It is less computational expensive and complicated to maximize a sum compared to a product. Furthermore, we can make use of ln(ab)=ln(a)+ln(b). Now we can maximize simply by taking derivatives and setting equal to 0.

```
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)
```

```
likelihood_function(m5.1.1, df_exp1$right_answer)
```

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

```
log_likelihood_function(m5.1.1, df_exp1$right_answer)
```

```
## [1] -10563.5
```

```
logLik(m5.1.1)
```

```
## 'log Lik.' -10622.03 (df=4)
```

You can see the difference between -10563.5 (our function estimate) and another build in function logLik = -10622.03.

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.

```
m5.2.1 <- glm(right_answer ~ 1, family = binomial(link = "logit"), data = df_exp1)
m5.2.2 <- glm(right_answer ~ target.frames , family = binomial(link= "logit"), data =
df_exp1)
m5.2.3 <- glmer(right_answer ~ target.frames + (1|subject), family = binomial(link =
"logit"), data = df_exp1)

m5.2.4 <- glmer(right_answer ~ target.frames  + (1+ target.frames|subject), family =
 binomial(link = "logit"), data = df_exp1)
summary(m5.2.4)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: right_answer ~ target.frames + (1 + target.frames | subject)
##    Data: df_exp1
##
##      AIC      BIC   logLik deviance df.resid
##  20907.7  20948.3 -10448.8  20897.7    25039
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -16.1429   0.0812   0.2454   0.4868   1.2632
##
## Random effects:
##  Groups  Name          Variance Std.Dev. Corr
##  subject (Intercept)   0.06086  0.2467
##          target.frames 0.05117  0.2262   -0.87
## Number of obs: 25044, groups:  subject, 29
##
## Fixed effects:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.09010    0.05878  -18.55   <2e-16 ***
## target.frames  0.83317    0.04433   18.80   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## target.frms -0.817
```

Following the guideline of the lme4 creators approach to selecting correlated or uncorrelated random effects in your model. https://cran.r-project.org/web/packages/lme4/vignettes/lmer.pdf (https://cran.r-project.org/web/packages/lme4/vignettes/lmer.pdf) (page. 7) The correlation

between target.frames and intercept is fairly high (-0.817) therefore this needs to be specified in our model by specifying (x|y). This isn't assessed by log-likelihood but rather correlation test between slope estimates and intercept.

```
anova( m5.2.4, m5.2.3, m5.2.1, m5.2.3)
```

	n...	AIC	BIC	logLik	deviance	Chisq	Df
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
MODEL3	1	26685.08	26693.21	-13341.54	26683.08	*NA*	*NA*
MODEL2	3	21250.06	21274.45	-10622.03	21244.06	5439.0149	2
MODEL4	3	21250.06	21274.45	-10622.03	21244.06	0.0000	0
MODEL1	5	20907.65	20948.29	-10448.83	20897.65	346.4127	2

4 rows

A model calculating the log-likelihoood ratio test. Since all our models are nested I've conditioned the order of variables following log-likelihood. If models weren't nested I would need to order by the amount of columns in the design matrix (X+Z).

```
log_like_ratio_test <- function(model1, model2){
  ratio = ifelse( logLik(model2) <= logLik(model1) , -2*(logLik(model2) - logLik(mode
l1)) , (-2*(logLik(model1) - logLik(model2)))) #log-like-ratio

  df_val = abs(df.residual(model1) - df.residual(model2)) #calculate the degree of fr
eedom
  p_val = pchisq(ratio, df = df_val, lower.tail = FALSE) #chi-square test
  name = deparse(substitute(c(model1, model2)))
  return(c(model_comp = name, log_lik_ratio = ratio, p_value = p_val))
}


log_like_ratio_test(m5.2.4, m5.2.1)
```

```
##          model_comp       log_lik_ratio             p_value
## "c(m5.2.4, m5.2.1)"  "5785.42764596847"                 "0"
```

```
log_like_ratio_test(m5.2.4, m5.2.2)
```

```
##              model_comp           log_lik_ratio                 p_value
##     "c(m5.2.4, m5.2.2)"      "832.846524015516" "3.25400522669055e-180"
```

```
log_like_ratio_test(m5.2.4, m5.2.3)
```

```
##             model_comp          log_lik_ratio                 p_value
##    "c(m5.2.4, m5.2.3)"     "346.412703775855" "5.99014236897163e-76"
```

```
#With package
pacman::p_load(lmtest)
lrtest(m5.2.3 , m5.2.4)
```

	#Df	LogLik	Df	Chisq	Pr(>Chisq)
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	3	-10622.03	*NA*	*NA*	*NA*
2	5	-10448.83	2	346.4127	5.990142e-76

2 rows

As shown, my functions output is similar to that of the packages lmtest::lrtest().

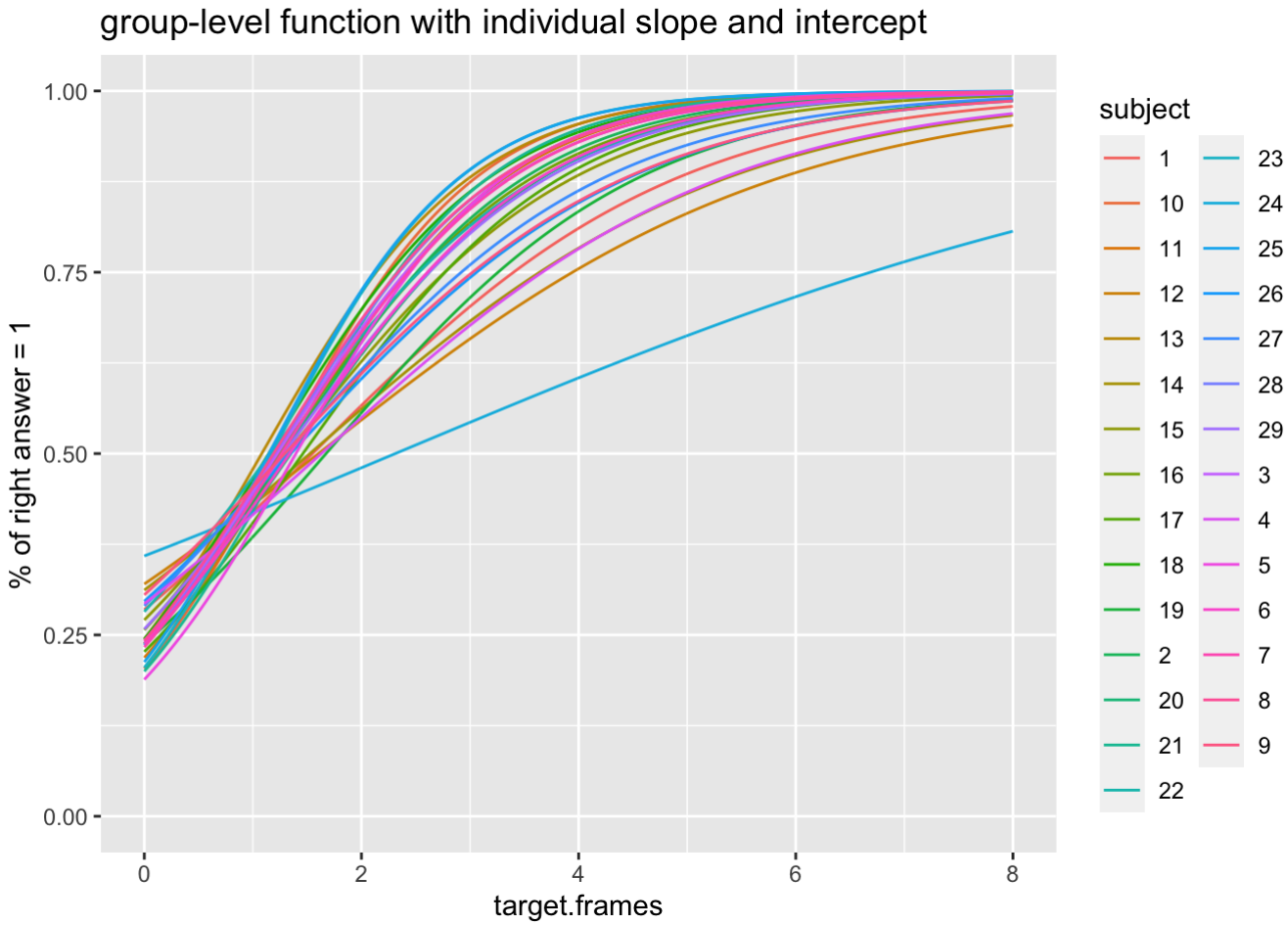
```
i. write a short methods section and a results section where you indicate which model
you chose and the statistics relevant for that choice. Include a plot of the estimate
d group-level function with `xlim=c(0, 8)` that includes the estimated subject-specif
ic functions.
```

logLik: We performed a log-likelihood ratio test on our nested models with the most complex model as reference level. With a significance level of 0.05 we will reject the null-hypothesis on all comparisons. This supports the choice of selecting the most complex ie. model4.

AIC: While log-likelihood ratio does account for model complexity by chi-square distribution varying depending on Df. Aikai-Information-Criteria (AIC) has shown to be a better measure for penalizing model complexity. the single level intercept model has the lowest AIC score and will therefore be the preferred model following AIC.

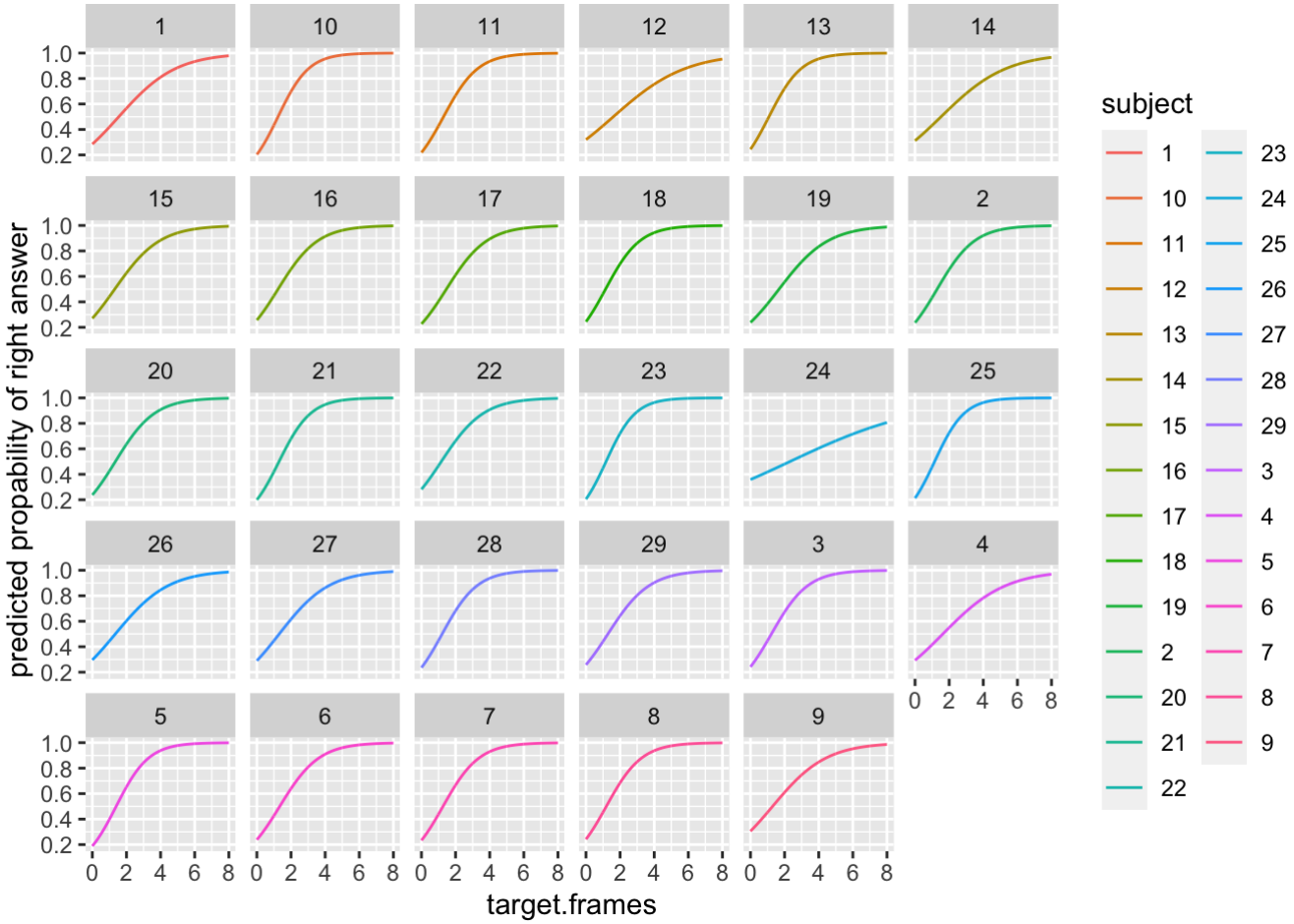
```
newdata1 <- data.frame(target.frames = rep(seq(0,8,0.01),29) , subject = as.character
(rep(seq(1:29),801))) %>%
  mutate(subject = as.factor(subject))
#predict values for each subject target.frames 0:8
newdata1$yhat <- predict(m5.2.4, newdata = newdata1, type = "response")
```

```
ggplot(newdata1, aes(x = target.frames, y = yhat, color = subject)) + geom_line() +
  ylim(0,1) + labs(title = "group-level function with individual slope and intercept"
, y = "% of right answer = 1")
```



While this plot is well suited for looking at the variance between subjects slope and intercept it is difficult to distinguish subjects from one another. If we facet by subject it will allow individual inspection.

```
ggplot(newdata1, aes(x = target.frames, y = yhat)) + geom_line(aes(colour = subject)) + labs(y = "predicted propability of right answer") +  facet_wrap(~subject) + xlim(0, 8)
```



While graphs are nice numbers can also be very informative in illustrating the spread in our beta parameters b0 and b1.

```
#get coef
coef_m5.2.4 <- coef(m5.2.4)
slope <- invlogit(coef_m5.2.4$subject[,2])
intercept <- invlogit(coef_m5.2.4$subject[,1])
cbind(intercept = summary(intercept), slope = summary(slope))
```

```
##         intercept     slope
## Min.    0.1885187 0.5624312
## 1st Qu. 0.2332623 0.6643165
## Median  0.2419339 0.7031253
## Mean    0.2543926 0.6944146
## 3rd Qu. 0.2850686 0.7278806
## Max.    0.3588732 0.7600487
```

Supported by the plots and statistical summary we can can conclude that there isn't that much spread in the individual variance at subject level.

```
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 %)
```

```
pacman::p_load(caret)
```

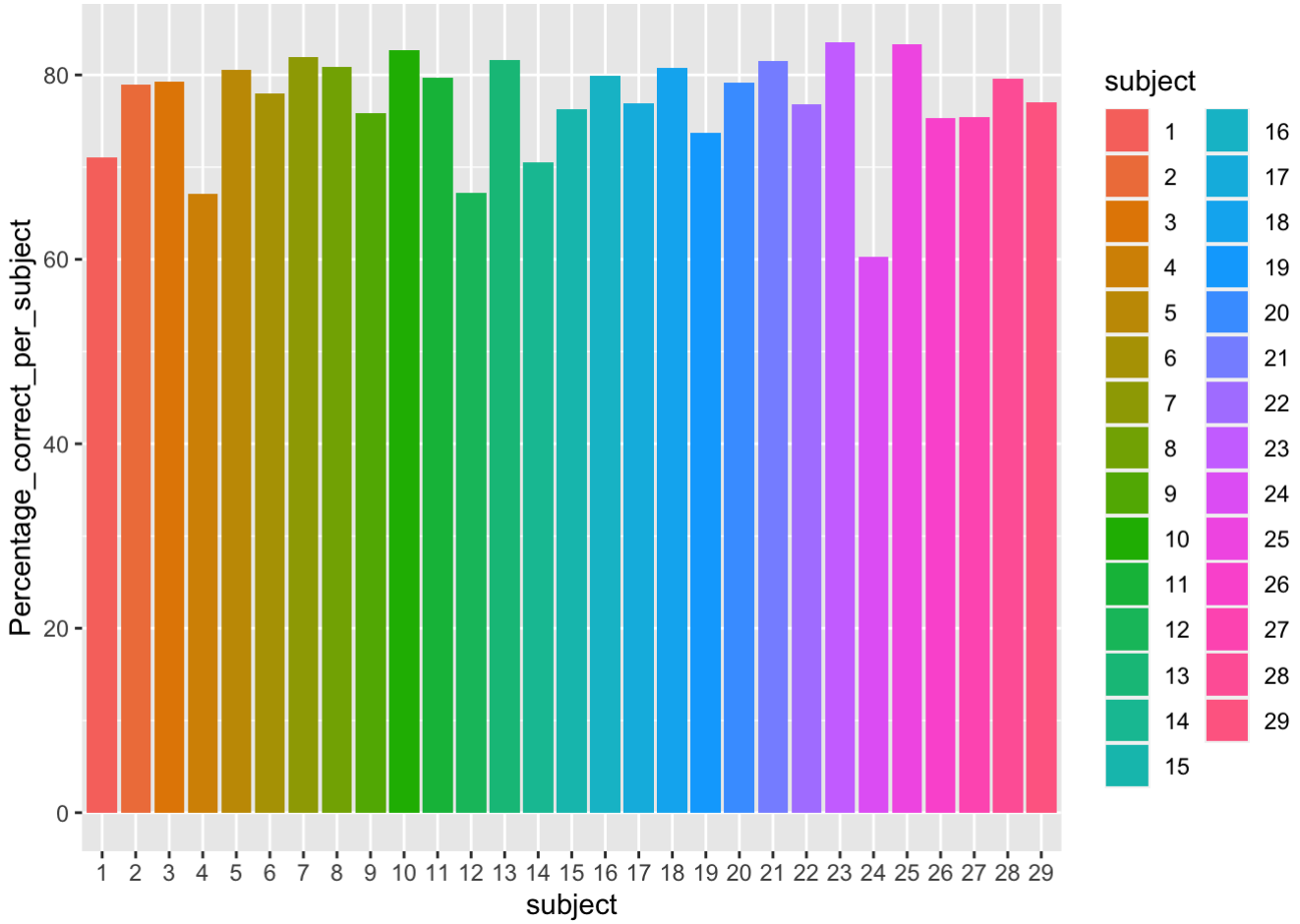
Generating an idea of accuracy in the entire data-set.

```
df_exp1<- df_exp1 %>%
  mutate(fitted_m.5.2.4 = fitted(m5.2.4)) %>%
  mutate(bin_fit_m5.2.4 = if_else(fitted_m.5.2.4 >= 0.5, 1, 0))

confusionMatrix(data = as.factor(df_exp1$bin_fit_m5.2.4), reference = as.factor(df_ex
p1$right_answer))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0  2101  2219
##          1  3525 17199
##
##                Accuracy : 0.7706
##                  95% CI : (0.7654, 0.7758)
##     No Information Rate : 0.7754
##     P-Value [Acc > NIR] : 0.9634
##
##                   Kappa : 0.2825
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.37344
##             Specificity : 0.88572
##          Pos Pred Value : 0.48634
##          Neg Pred Value : 0.82991
##              Prevalence : 0.22464
##          Detection Rate : 0.08389
##    Detection Prevalence : 0.17250
##       Balanced Accuracy : 0.62958
##
##        'Positive' Class : 0
##
```

We illustrate differences in subject accuracy

```
df_exp_temp <- df_exp1 %>%
  group_by(subject) %>%
  summarise(Percentage_correct_per_subject = sum(bin_fit_m5.2.4 == right_answer)/leng
th(right_answer)*100)

ggplot(df_exp_temp, aes(x = subject, y = Percentage_correct_per_subject, fill = subje
ct)) + geom_col()
```



Based on a visual inspection of predicted correct answers we can see that subject 24 has the worst predicted accuracy score. We will therefore perform as one-sample test to see if he performs better than chance.

```
df_exp1_fil24 <- df_exp1 %>%
  filter(subject == 24)

#t-test on collected data
t.test(as.numeric(df_exp1_fil24$right_answer), mu = 0.5, alternative = "greater")
```

```
##
##  One Sample t-test
##
## data:  as.numeric(df_exp1_fil24$right_answer)
## 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
```

```
#t-test on predicted values.
t.test(as.numeric(df_exp1_fil24$bin_fit_m5.2.4), mu = 0.5, alternative = "greater")
```

```
##
##  One Sample t-test
##
## data:  as.numeric(df_exp1_fil24$bin_fit_m5.2.4)
## t = 10.473, df = 873, p-value < 2.2e-16
## alternative hypothesis: true mean is greater than 0.5
## 95 percent confidence interval:
##  0.6407847       Inf
## sample estimates:
## mean of x
## 0.6670481
```

T.test Summary

collected data The accuracy for subject 23 were above chance (50%), (t(863) = 26.944, p = 2.2e^16)

predicted data The predicted accuracy for subject 23 were above chance (50%), (t(863) = 26.275, p = 2.2e^16)

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

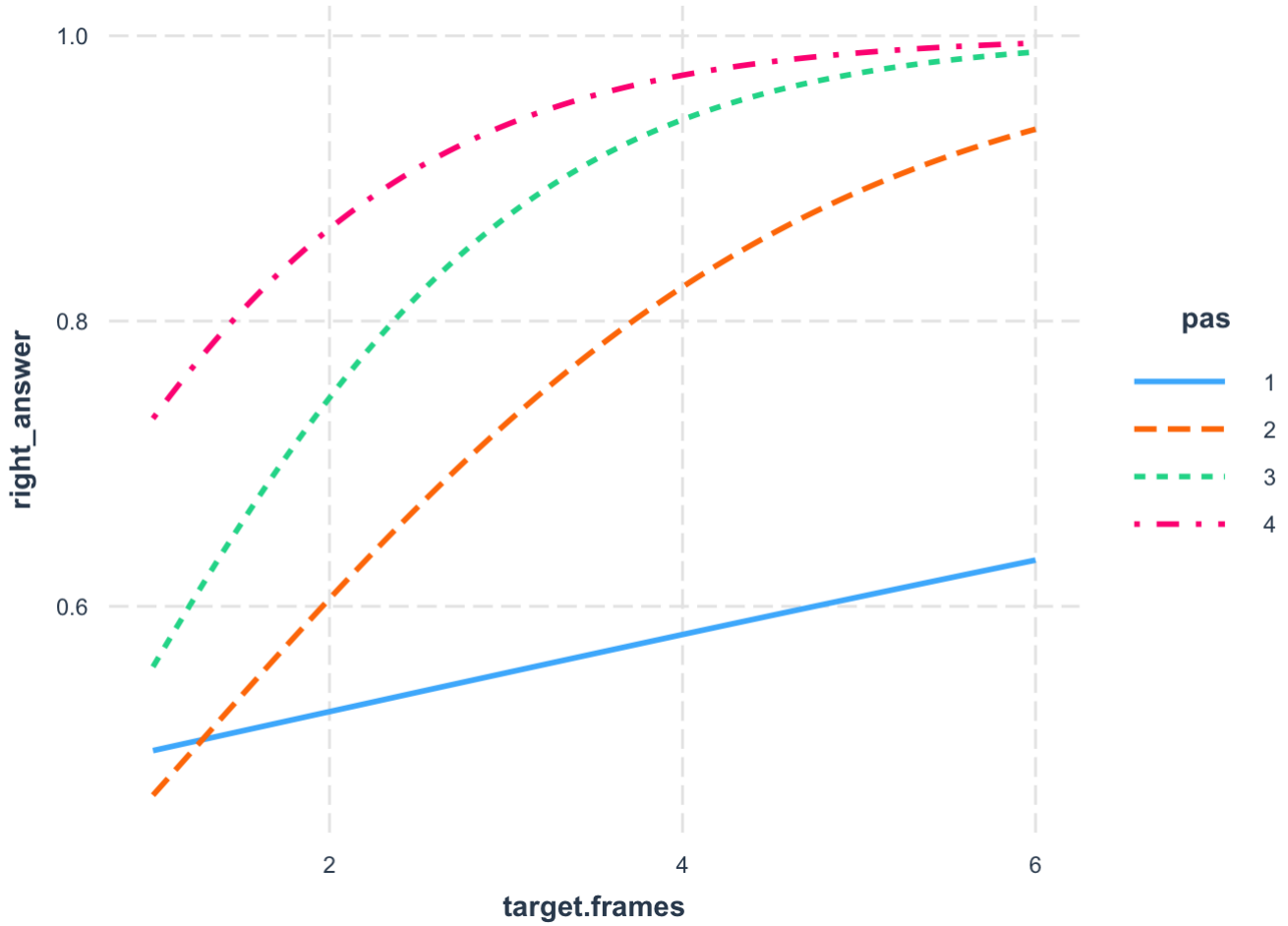
```
m5.3.1 <- glmer(right_answer ~ pas*target.frames + (target.frames||subject), data = d
f_exp1, family = binomial(link = "logit"), control = glmerControl(optimizer="bobyqa"
))



log_like_ratio_test(m5.3.1 , m5.2.4)
```

```
##              model_comp           log_lik_ratio                 p_value
##     "c(m5.3.1, m5.2.4)"      "1405.89148846487" "7.28140374467874e-302"
```

Our log-likelihood-ratio test support the choice of the more complex model. (m5.3.1) including the interaction between pas:target.frames and pas as a fixed effect.

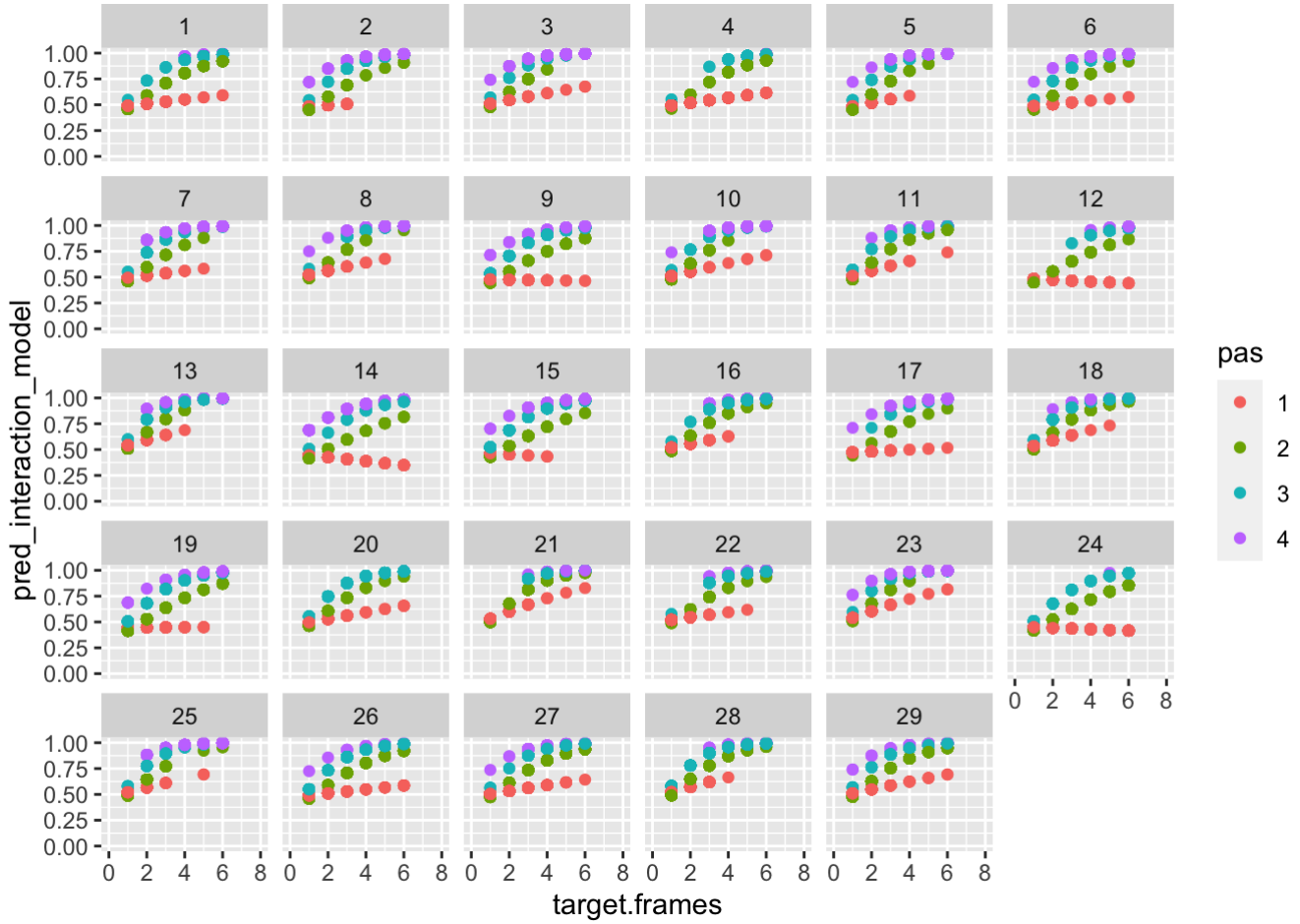
```
ii. plot the estimated group-level functions over `xlim=c(0, 8)` for each of the four
PAS-ratings - add this plot to your report (see: 5.2.i) and add a description of your
chosen model. Describe how _pas_ affects accuracy together with target duration if at
all. Also comment on the estimated functions' behaviour at target.frame=0 - is that b
ehaviour reasonable?
```

```
df_exp1 <- df_exp1 %>%
  mutate(pred_interaction_model = fitted.values(m5.3.1))


interactions::interact_plot(model = m5.3.1 , pred = "target.frames", modx = "pas")
```



While this plot is nice for showing the general interaction between pas:target.frames it does not account for our random intercept and random slop of target.frames. We will therefore continue with another plot that can illustrate these individual differences.

```
ggplot(df_exp1, aes(x = target.frames, y = pred_interaction_model, col = pas )) + geo
m_point() + facet_wrap(~subject) + xlim(0,8) + ylim(0,1)
```



As shown target.frames has individual # 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.

```
coef_OG_scale <-  data.frame(intercept = rep(NA,29), pas2 = rep(NA,29), pas3 = rep(NA
,29), pas4 = rep(NA,29), target.frames = rep(NA,29), pas2_target.frames = rep(NA,29),
pas3_target.frames = rep(NA,29), pas4_target.frames = rep(NA,29))

for (i in 1:length(coef(m5.3.1)$subject)){
  coef_OG_scale[,i] = invlogit(coef(m5.3.1)$subject[,i])
}
```

All beta values were transformed from log-odds to probabilities through an inverse logit transformation. All beta values will be reported on the probability scale for easier interpretation. ’

Intercepts Pas is coded as a factor and we will therefore gain an intercept for all 4 levels of pas. *general baseline intercept* $_0 (intercept/pas1) = 0.4714, z = -2.064, p = 0.0390 $ *Individual baseline intercept per subject* $\beta_0(intercept/pas1) : Median = 0.47, Mean = 0.4716, sd = 0.011$ Every subject has their own baseline/intercept probabillity of giving the right answer when target.frames = 0 and pas = pas1.

$\beta_1(intercept/pas2) = 0.3604, z = -6.471, p = 9.77e-11$ added on top of of their individual baseline for pas1
$\beta_2(intercept/pas3) = 0.3771, z = -3.653, p = 0.000259 * **$ added on top of of their individual baseline for pas1
$\beta_3(intercept/pas4) = 0.5658, z = 1.065, p = 0.286944$ added on top of of their individual baseline for pas1

Slopes We’ve modeled an interaction between a categorical variable(pas1-4) and a numeric (target.frames) with a second level effect of random slope We will therefore have a individual slope of target.frame for each subject. **main effect** $\beta_4(target.frames) = Median = 0.5282, Mean = 0.5269, sd = 0.0234$ **interaction effect** $_5 (target.frames:pas2) = 0.6102, z = 13.006, p < 2e-16 $ $_6 (target.frames:pas3) = 0.6764, z = 16.336, p < 2e-16 $ $_7 (target.frames:pas4) = 0.6776, z = 10.990, p < 2e-16 $

Currently both genereal and individual intercept are below chance (50%). An explanation can be found in the interpritation. Currently our intercept represent target.frames = 0 and pas = pas1. Having been presented with 0 frames does not make much sense in terms of the experiment. But we would expect a 50/50 chance to guess right if you had no information to guess by. The best solution would be to standardize the data around target.type. Giving us a more intuitive interpritation.

Our model supports our assumption of an increase in subjective (pas-score) and objective (target.frames) evidence significantly increases the probability of the right answer.

Furthermore, as shown by our interaction effects increasing subjective evidence (PAS1 -> PAS2/3/4) will significantly modulate objective evidence to increase the accuracy score even faster compared to PAS1.

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

```
design <- model.matrix(m5.3.1)
head(design)
```

```
##   (Intercept) pas2 pas3 pas4 target.frames pas2:target.frames
## 1           1    0    0    0             3                   0
## 2           1    0    0    0             2                   0
## 3           1    0    0    0             1                   0
## 4           1    0    1    0             5                   0
## 5           1    0    1    0             6                   0
## 6           1    0    0    1             4                   0
##   pas3:target.frames pas4:target.frames
## 1                  0                  0
## 2                  0                  0
## 3                  0                  0
## 4                  5                  0
## 5                  6                  0
## 6                  0                  4
```

```
pacman::p_load(multcomp)
contrast.vectors <- matrix(c(0, 0, 0, 0, -1, 1, 0, 0), nrow=1)
ghs <- glht(m5.3.1, contrast.vectors)
print(summary(ghs))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: glmer(formula = right_answer ~ pas * target.frames + (target.frames ||
##     subject), data = df_exp1, family = binomial(link = "logit"),
##     control = glmerControl(optimizer = "bobyqa"))
##
## Linear Hypotheses:
##        Estimate Std. Error z value Pr(>|z|)
## 1 == 0  0.33877    0.05723   5.919 3.24e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

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

```
#Contrast matrix


contrast_matrix_compar <- glht(m5.3.1, linfct = c( "pas2:target.frames = -1", "pas3:t
arget.frames = 1"))
summary(contrast_matrix_compar) #not a 100% how this works.
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: glmer(formula = right_answer ~ pas * target.frames + (target.frames ||
##     subject), data = df_exp1, family = binomial(link = "logit"),
##     control = glmerControl(optimizer = "bobyqa"))
##
## Linear Hypotheses:
##                           Estimate Std. Error z value Pr(>|z|)
## pas2:target.frames == -1  0.44820    0.03442   42.07  < 1e-10 ***
## pas3:target.frames == 1   0.73767    0.04507   -5.82 1.18e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

```
#Difference between pas2:target.frames pas3:target.frames
contrast.vector <- matrix(c(0, 0, 0, 0, 0, -1, 1, 0), nrow=1)
gh <- glht(m5.3.1, contrast.vector)
print(summary(gh))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: glmer(formula = right_answer ~ pas * target.frames + (target.frames ||
##     subject), data = df_exp1, family = binomial(link = "logit"),
##     control = glmerControl(optimizer = "bobyqa"))
##
## Linear Hypotheses:
##        Estimate Std. Error z value Pr(>|z|)
## 1 == 0  0.28947    0.04578   6.323 2.56e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

objective evidence is also modulated more by PAS3 than PAS2 resulting in a faster growing accuracy by target.frames in PAS3 and than PAS2.

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

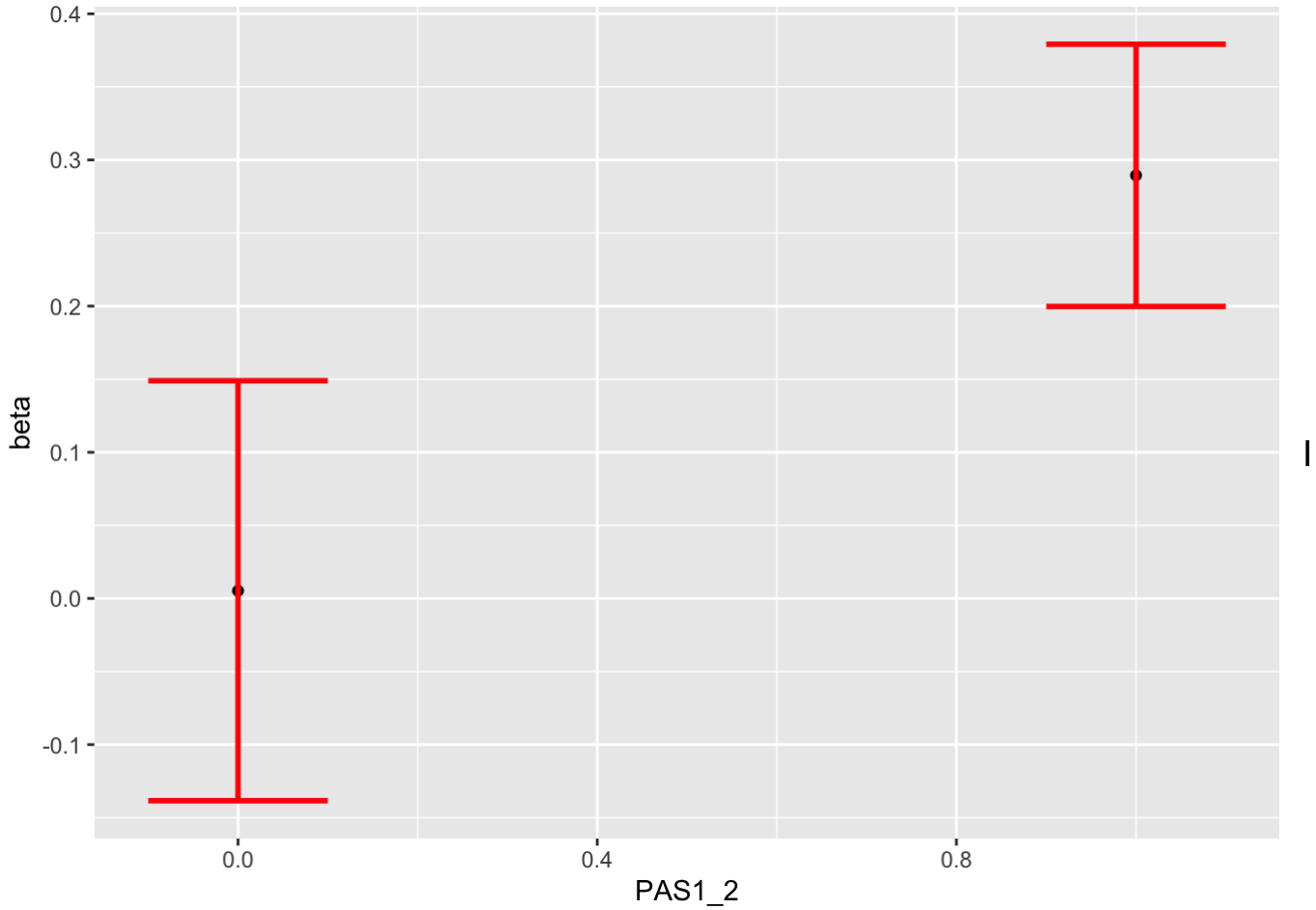
```
#Difference between pas3:target.frames pas4:target.frames
contrast.vector2 <- matrix(c(0, 0, 0, 0, 0, 0, -1, 1), nrow=1)
gh2 <- glht(m5.3.1, contrast.vector2)
print(summary(gh2))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: glmer(formula = right_answer ~ pas * target.frames + (target.frames ||
##     subject), data = df_exp1, family = binomial(link = "logit"),
##     control = glmerControl(optimizer = "bobyqa"))
##
## Linear Hypotheses:
##        Estimate Std. Error z value Pr(>|z|)
## 1 == 0 0.005259   0.073293   0.072    0.943
## (Adjusted p values reported -- single-step method)
```

Moving from PAS3-PAS4 does not increase the effect of target.frames significantly.

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)

```
# beta = 0.44821   std.error =  0.03446 pas1-pas2
# beta = 0.005258   std.error = 0.073618 pas3-pas4
confint_gh<- confint(gh)
confint_gh2 <- confint(gh2)

df_confint <- data.frame(beta = c(confint_gh$confint[1], confint_gh2$confint[1]),
           upper = c(confint_gh$confint[2], confint_gh2$confint[2]),
           lower = c(confint_gh$confint[3], confint_gh2$confint[3]),
           PAS1_2 = c(1,0),
           PAS3_4 = c(0,1))

ggplot(df_confint, aes(x = PAS1_2, y = beta)) + geom_point() + geom_errorbar(aes(x =
 PAS1_2, ymin = lower, ymax = upper), width = 0.2, color = "red", size = 1)
```



can not do a t-test as calculating SE_pooled requires the SD.
$SE_p = SD_p * sqrt(1/n_1 + 1/n_2)$. We can calculate the 95% confidence intervals based on SE x 1.96 for lower and high boundaries. Our Confidence intervals does not overlap which would indicate that they are significantly different. THe difference of **PAS1-2** is larger than **PAS3-4**.

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

    ## "par" are our four parameters (a numeric vector)
    ## par[1]=a, par[2]=b, par[3]=c, par[4]=d
    a = par[1]
    b = par[2]
    c = par[3]
    d = par[4]
    x <- data$x
    y <- data$y
    y.hat <- a + ((b-a)/(1+(exp((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)

```
#Optim function with target.frames
data <- df_exp1 %>%
  dplyr::select(target.frames, right_answer) %>%
  rename(x = target.frames, y = right_answer)


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

```
## $par
## [1] 0.5077512 0.9586143 2.9380622 0.4303382
##
## $value
## [1] 3500.344
##
## $counts
## function gradient
##       24       24
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

```
#Optim function with pas rating.
data1 <- df_exp1 %>%
  filter(subject == 7) %>% #only use subject 7
  dplyr::select(pas, right_answer) %>%
  mutate(pas = as.numeric(pas)) %>% #optim function requires it to be numeric.
  rename(x = pas, y = right_answer)


optim_pas <- optim(par = c(0.5,1,1,1), fn = RSS, data = data1, method = "L-BFGS-B", l
ower = c(0.5,0.5,-Inf,-Inf),
      upper = c(1,1,Inf,Inf))
optim_pas$par
```

```
## [1] 0.5000000 0.9945230 2.6256293 0.3288213
```

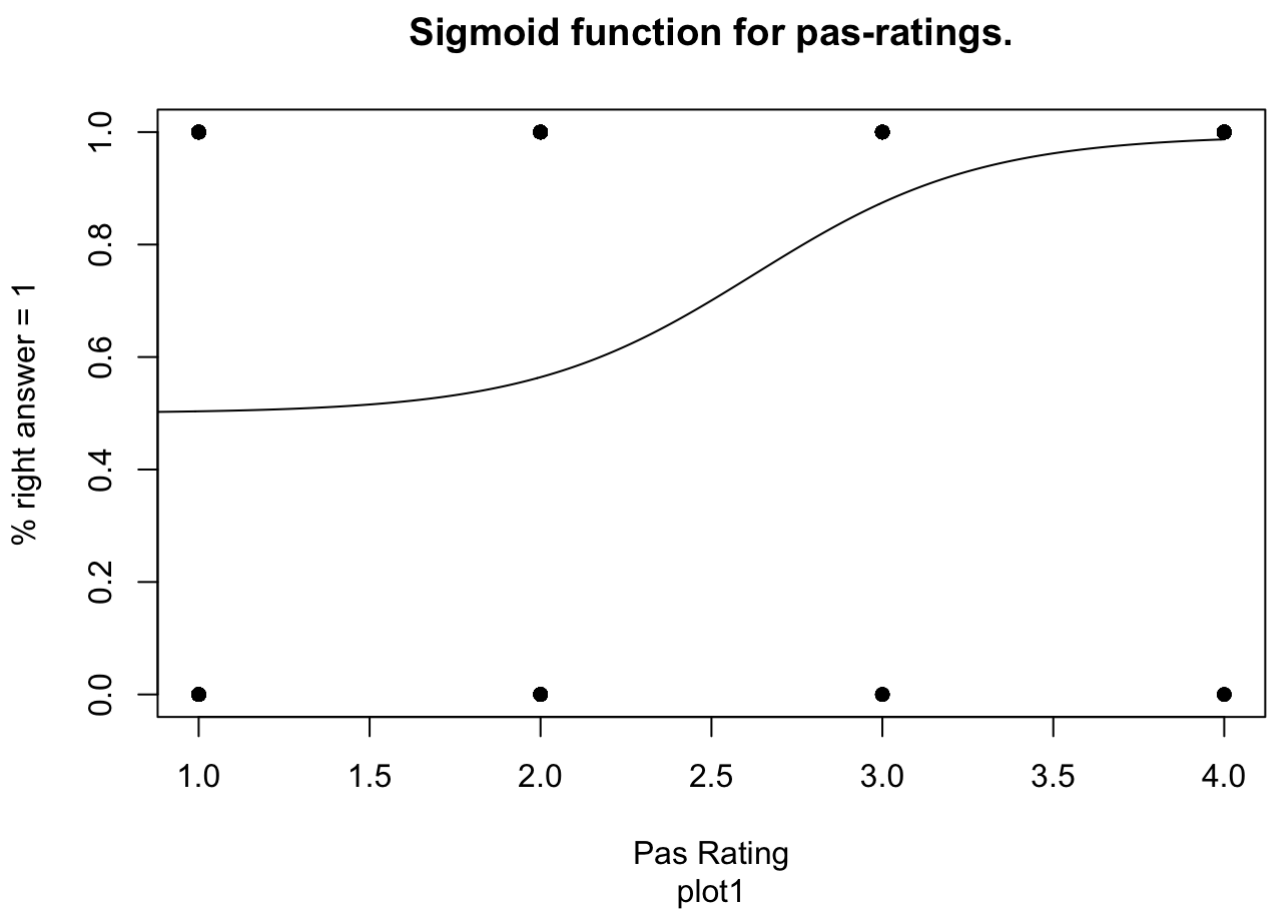
par = c(0.5, 1,1,1,) *a* will start at 0.5 with a min = 0.5 and max = 1. *b* will start at 1 with min = 0.5 and max = 1. The choice of min and max is based on the assignments wish to avoid values below chance. So max and min values of y can span from 0.5:1. It is imaginable that such values could be possible so therefore they have been chosen as the lower and upper boundraies.

Starting point is based on guesses of what the best scenario could be and worst scenario. ie. a = 0.5 and b = 1.

Use the estimated parameters to compute y_hat and plot the function.

```
sigmoid <- function(a,b, c,d,x){
  y = a + ((b-a)/(1+(exp((c-x)/d))))
  return(y)
}

x_pas_optim <- seq(0, 4, 0.01)
y_pas_optim <- sigmoid(optim_pas$par[1],optim_pas$par[2],
                       optim_pas$par[3],optim_pas$par[4],x_pas_optim)
```

```
plot(data1$x, data1$y, pch = 16, xlab = "Pas Rating", ylab = "% right answer = 1")
lines(x_pas_optim, y_pas_optim) + title(main = "Sigmoid function for pas-ratings.", s
ub = "plot1")
```




```
## integer(0)
```

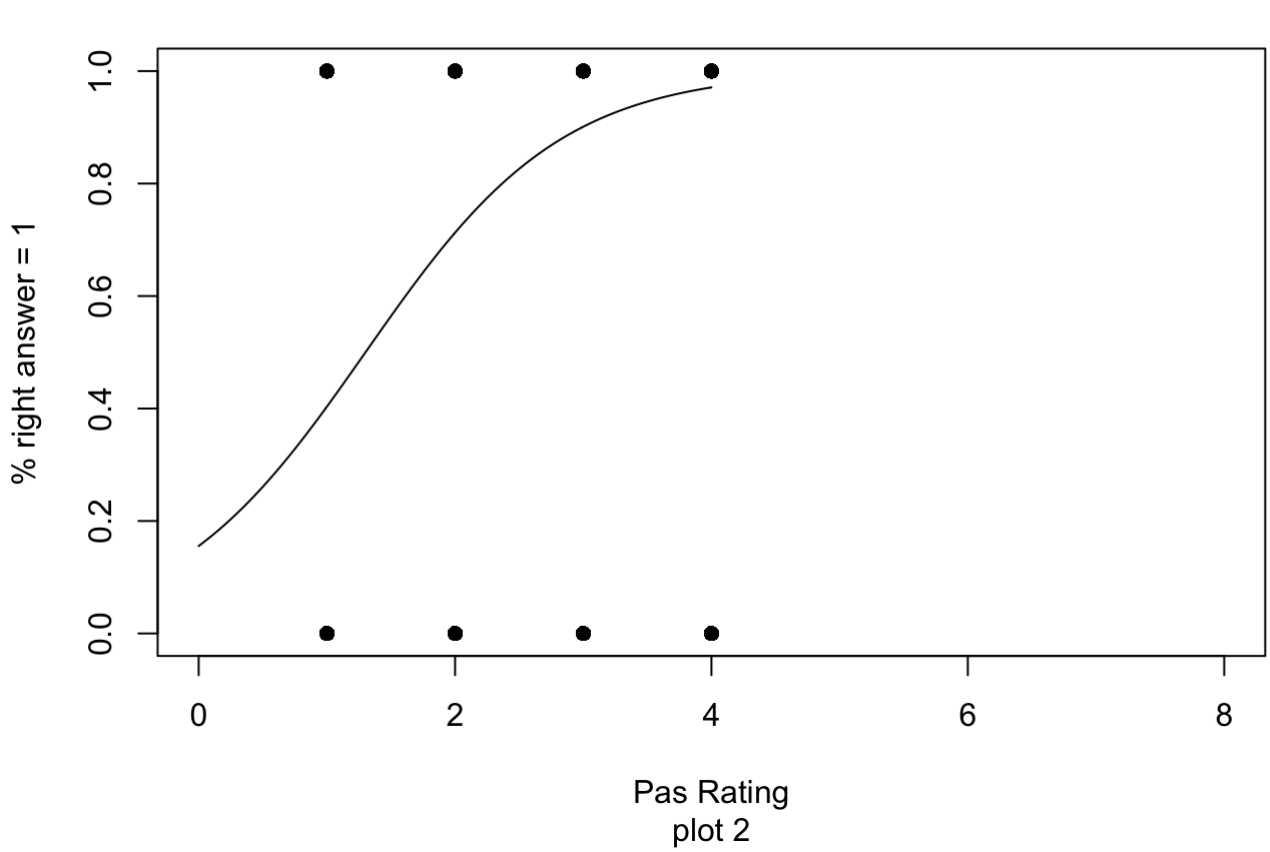
ii. Plot the fits for the PAS ratings on a single plot (for subject 7) `xlim=c(0, 8)`

```
#Simple model to do illustrate pas ratings for subject 7
m7.1.1 <- glm(right_answer ~ as.numeric(pas), data = filter(df_exp1, subject == 7), f
amily = binomial(link = "logit"))
x_pas <- seq(0, 4, 0.01)
y_pas <- predict(m7.1.1, list(pas = x_pas),type="response")
```

```
#plots for subject 7
plot(as.numeric(df_exp1$pas), df_exp1$right_answer, pch = 16, xlab = "Pas Rating", yl
ab = "% right answer = 1",
     xlim = c(0,8)) + title(main = "subject 7 glm function with no interaction", sub
 = "plot 2")
```

```
## integer(0)
```

```
lines(x_pas, y_pas)
```



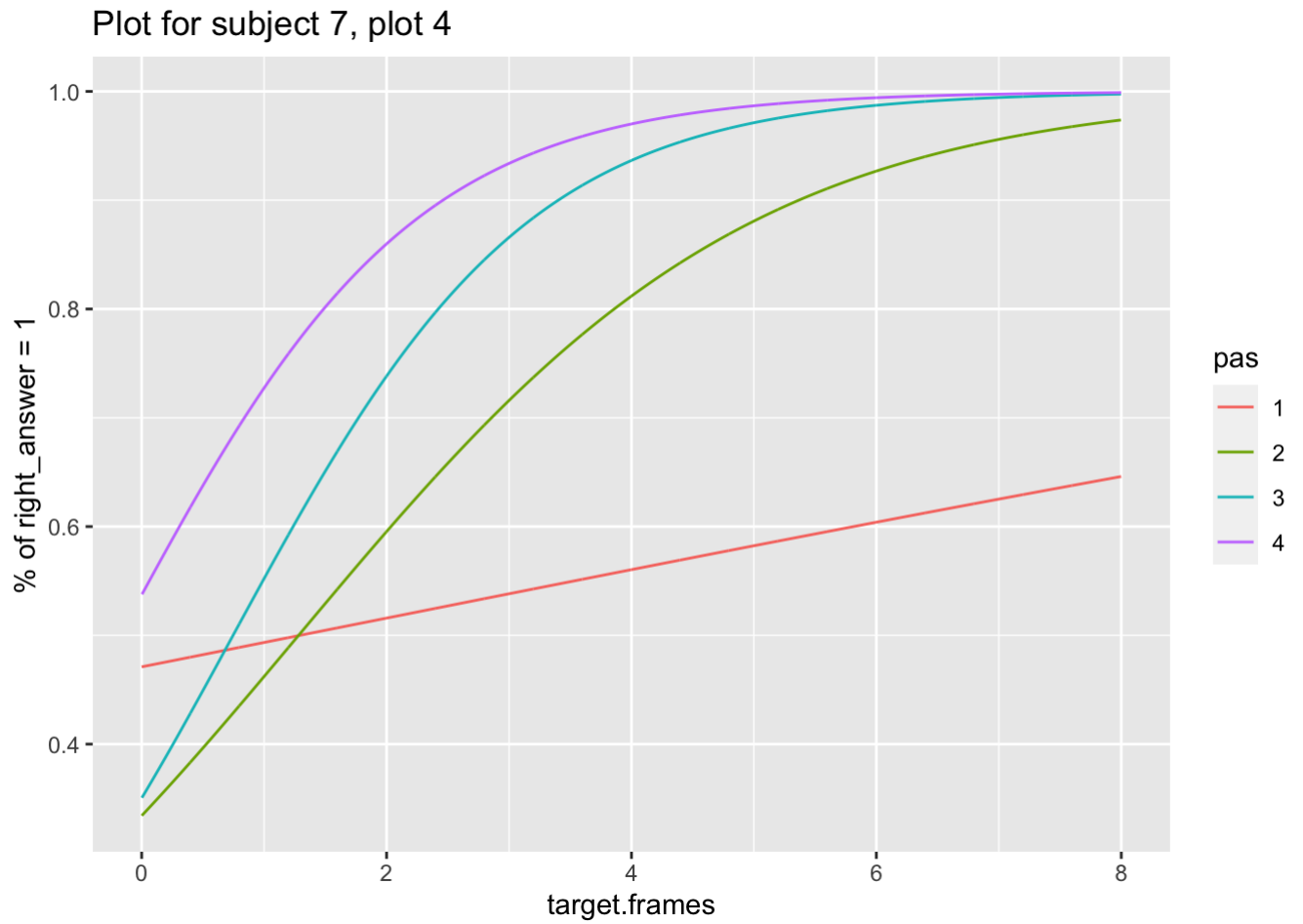
iii. Create a similar plot for the PAS ratings on a single plot (for subject 7), but this time based on the model from 6.1 `xlim=c(0, 8)`

```
newdat <- data.frame(cbind('target.frames' = seq(0, 8, by = 0.001), 'pas' = rep(1:4), 'subject' = rep('7')))
```

```
## Warning in cbind(target.frames = seq(0, 8, by = 0.001), pas = rep(1:4), : number
## of rows of result is not a multiple of vector length (arg 2)
```

```
newdat$subject <- as.factor(newdat$subject)
newdat$pas <- as.factor(newdat$pas)
newdat$target.frames <- as.numeric(newdat$target.frames)

newdat$yhat <- predict(m5.3.1, newdata = newdat, type = "response")
```

```
ggplot(newdat, aes(x = target.frames, y = yhat, color = pas)) + geom_line() + labs(title = "Plot for subject 7, plot 4", y = "% of right_answer = 1")
```




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

Comparing plot1 (sigmoid) and plot2 (subject 7): Plot 1 shows that as PAS approaches 4 the probability of guessing right approaches 100%. This is also illustrated in **plot 3** but also adds the detail of how PAS4 interacts with target.frames. While higher level of target.frames along with PAS4 rating will result in the best probability of guessing right.

While **plot 1** gives us the best idea of how PAS rating effects probability of guessing correct. **plot 3** is more informative when it comes to illustrating the combination of target.frames and PAS. But as we will show in the next exercise there is a way to combine the information from both plots into 1 single 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))

    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.

```
#Create a function with a loop for calculating a,b,c and d for every subject within e
very level of PAS.

optim_for_indi <- 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 <- df_exp1 %>%
      filter(pas == i)

    for (ii in 1:length(unique(df_exp1$subject))){
      data_temp <- df_temp1 %>%
        filter(subject == ii) %>%
        dplyr::select(target.frames, right_answer, pas) %>%
        rename(x = target.frames, y = right_answer)

      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)
}
```

```
#Get all parameters for each subject in each level of pas
df_param <- optim_for_indi() %>%
  na.omit()
df_param
```

	subject	...	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

	subject	...	a	b	c	d
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
10	9	1	0.5000000	0.5391376	1.7117410	0.0203760864
11	10	1	0.5000000	0.5000000	1.0000000	1.0000000000

1-10 of 116 rows Previous **1** 2 3 4 5 6 ... 12 Next

```
#Compute the average value of a,b,c and d.
df_param_avg <- df_param %>%
  group_by(pas) %>%
  summarise(a = mean(a), b = mean(b), c = mean(c), d = mean(d))

df_param_avg
```

pas	a	b	c	d
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	0.5796434	0.5514403	1.638820	0.4571516
2	0.5274091	0.8775394	2.829774	0.1757673
3	0.5768916	0.9556266	2.281409	0.2259202
4	0.7199105	0.9658502	2.189910	0.5010169

4 rows

We currently have a problem of $a > b$ in PAS1. This will result in a declining sigmoid function. However, I can not specify logical statement in lower_boundaries ensuring $b > a$.

```
#x-values
x_target.frames <- seq(0,8,0.01)


#setup data frame for x and y of different PAS score.
df_x_y <- data.frame(x_target.frames)

#Function for estimating y_hat given our a,b,c and d from our optim function.
y_hat_func <- function(){
for (i in 1:4){
  y_hat_temp <- sigmoid(df_param_avg$a[i], df_param_avg$b[i], df_param_avg
                        $c[i], df_param_avg$d[i], x_target.frames)
  df_x_y[,i+1] <- y_hat_temp
}

  df_x_y <- df_x_y %>% #Rename column names into something meaningful.
    rename(x = x_target.frames, y_hat_pas1 = V2 , y_hat_pas2 = V3 , y_hat_pas3 = V4 ,
y_hat_pas4 = V5)
  return(df_x_y)
}

df_final_param <- y_hat_func()
```

```
head(df_final_param)
```

	x <dbl>	y_hat_pas1 <dbl>	y_hat_pas2 <dbl>	y_hat_pas3 <dbl>	y_hat_pas4 <dbl>
1	0.00	0.5788822	0.5274092	0.5769072	0.7229803
2	0.01	0.5788658	0.5274092	0.5769079	0.7230414
3	0.02	0.5788491	0.5274092	0.5769086	0.7231037
4	0.03	0.5788320	0.5274092	0.5769094	0.7231672
5	0.04	0.5788146	0.5274092	0.5769102	0.7232319
6	0.05	0.5787968	0.5274092	0.5769110	0.7232980

6 rows

```
tail(df_final_param)
```

	x <dbl>	y_hat_pas1 <dbl>	y_hat_pas2 <dbl>	y_hat_pas3 <dbl>	y_hat_pas4 <dbl>
796	7.95	0.5514403	0.8775394	0.9556266	0.9658477
797	7.96	0.5514403	0.8775394	0.9556266	0.9658478

	x	y_hat_pas1	y_hat_pas2	y_hat_pas3	y_hat_pas4
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
798	7.97	0.5514403	0.8775394	0.9556266	0.9658478
799	7.98	0.5514403	0.8775394	0.9556266	0.9658479
800	7.99	0.5514403	0.8775394	0.9556266	0.9658479
801	8.00	0.5514403	0.8775394	0.9556266	0.9658480

6 rows

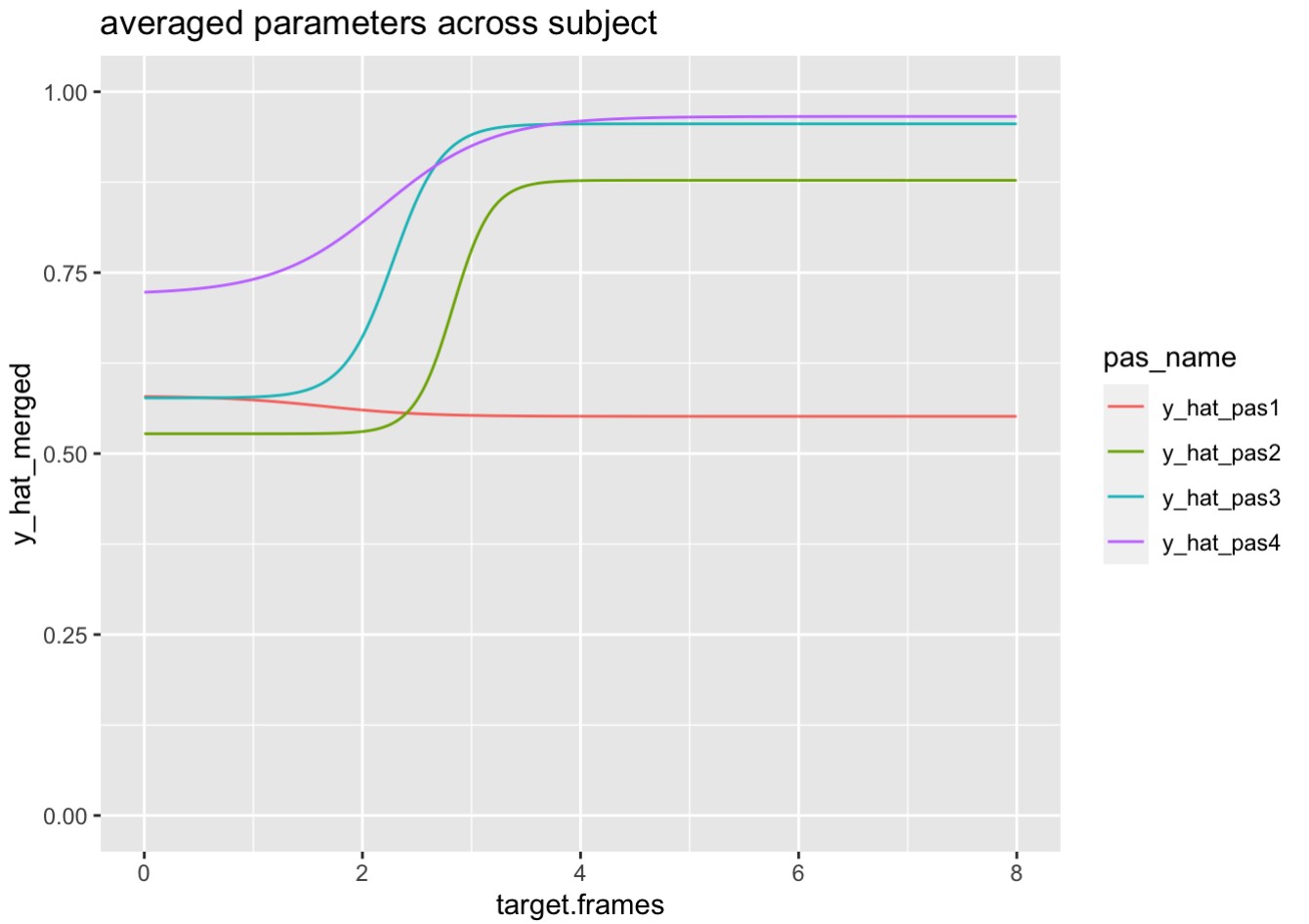
As X -> 8 our y_hat_pas1 estimates based on the sigmoid function defined in earlier chunks will decrease. This does not make sense what so ever but is due to *a* and *b* from our optim function.

```
df_final_param_long <- df_final_param %>%
  pivot_longer(cols = c(y_hat_pas1,y_hat_pas2,y_hat_pas3,y_hat_pas4) , names_to = "pa
s_name", values_to = "y_hat_merged")

df_final_param_long
```

	x	pas_name	y_hat_merged
	<dbl>	<chr>	<dbl>
	0.00	y_hat_pas1	0.5788822
	0.00	y_hat_pas2	0.5274092
	0.00	y_hat_pas3	0.5769072
	0.00	y_hat_pas4	0.7229803
	0.01	y_hat_pas1	0.5788658
	0.01	y_hat_pas2	0.5274092
	0.01	y_hat_pas3	0.5769079
	0.01	y_hat_pas4	0.7230414
	0.02	y_hat_pas1	0.5788491
	0.02	y_hat_pas2	0.5274092

1-10 of 3,204 rows Previous **1** 2 3 4 5 6 … 321 Next

```
ggplot(df_final_param_long, aes(x, y_hat_merged, colour = pas_name)) +
  geom_line() + ylim(0,1) + labs(x = "target.frames", title = "averaged parameters ac
ross subject")
```



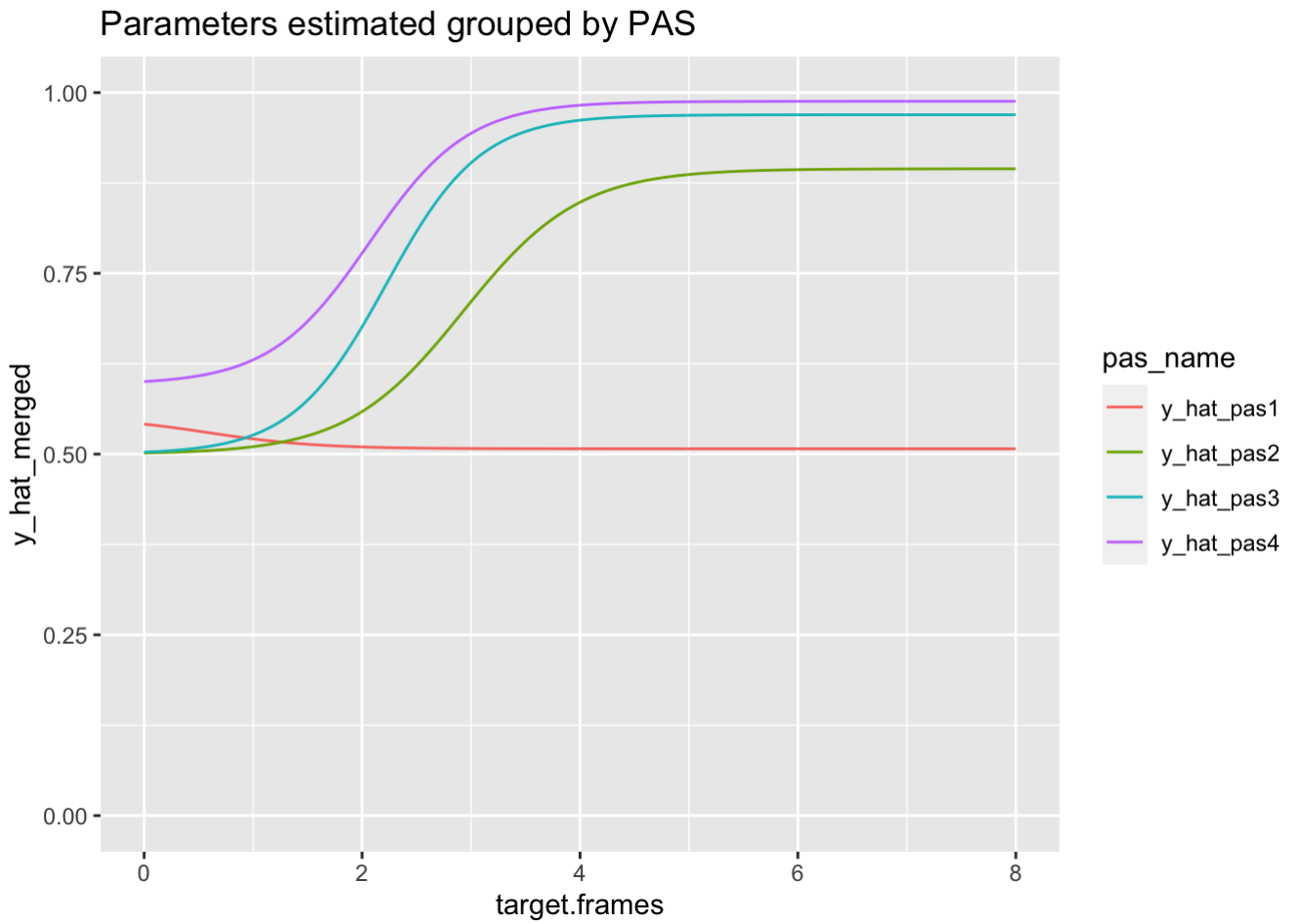
Extra stuff

Checking the difference between taking the mean() of every parameter c(a,b,c,d) and running the optim on data only divided by PAS ratings.

```
#Make function

optim_pas_overall <- function(i){
  #prepare correct subset of  data frame.
  pas_temp <- df_exp1 %>%
    filter(pas == i) %>%
    dplyr::select(target.frames, right_answer) %>%
    rename(x = target.frames, y = right_answer)
  op_temp_pas = optim(par = c(0.5,1,1,1), fn = RSS, data = pas_temp, method = "L-BFGS
-B", lower =  c(0.5,0.5,-Inf,-Inf), upper = c(1,1,Inf,Inf))

  # x and y values
  x_pas_optim <- seq(0, 8, 0.01)
  y_pas_optim <- sigmoid(op_temp_pas$par[1],op_temp_pas$par[2],
                         op_temp_pas$par[3],op_temp_pas$par[4],x_pas_optim)
  return(y_pas_optim)
}
```

```
#Generate y_hats with function for all pas ratings.
data_frame_y_hats <- tibble(x = x_target.frames, y_hat_pas1 = optim_pas_overall(1)
                            , y_hat_pas2 = optim_pas_overall(2)
                            , y_hat_pas3 = optim_pas_overall(3)
                            , y_hat_pas4 = optim_pas_overall(4))
```

```
#Pivot longer to merge into 1 columns with an identity column.
data_frame_y_hats_long <- data_frame_y_hats %>%
  pivot_longer(cols = c(y_hat_pas1,y_hat_pas2,y_hat_pas3,y_hat_pas4) , names_to = "pa
s_name", values_to = "y_hat_merged")

#plot
ggplot(data_frame_y_hats_long, aes(x, y_hat_merged, colour = pas_name)) +
  geom_line() + ylim(0,1) + labs(x = "target.frames", title = "Parameters estimated g
rouped by PAS")
```



Advantage If the relation between *% of guessing correct* and *target.frames* is in nature a sigmoid function. ie. When target.frames goes from 0 -> *inf* then would derivative first be fairly low (difference between 0-1 isn't that great) while at some point we would hit the inflection point (point with highest derivative and effect of increasing target.frame by any unit.) And afterwards the derivative would lessen because increasing target.frames above a specific number would not result in any new information and therefore better accuracy.

Disadvantage/problem This plot/method alows us to set a minimum of our y-values. I've specified lower boundary of a = 0.5 (not allowing below chance). As this was our goal with using the optim function. Though there are several arguments against setting a minimum y-value. (It's cheating. :P)

The Optim function tries to minimise our RSS given our other parameters (a,b,c,d). So if we really were to compare the model from optim with min = 0.5 and our interaction model from exercise 5 we should look at diff in RSS. Specifying absurd high/low as *a* & *b* = 1 would give incredible accuracy by may have inflated RSS and poor prediction accuracy of unseen data.