Portfolio Assignment 2, part 2, Methods 3, 2021, autumn semester

Study Group 8, Emma Rose Hahn (EH), Viara Krasteva (VK), Kristian Nøhr Villebro (KV), Luke Ring (LR)

2021-10-27

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)

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

Answers

Exercise 4, part 1 - EH

```
#loading in the data
dat <- list.files(path = "./experiment_1", full.names = TRUE)


#Changing all the variables classes to the ones we want them to be
samples <- map_df(dat, read_csv,
    trim_ws = TRUE, na = c("", "NA"), # i
    col_types = cols(
        trial.type = col_factor(),
        pas = col_factor(),
        trial = col_factor(),
        jitter.x = col_double(),
        jitter.y = col_double(),
        odd.digit = col_integer(),
        target.contrast = col_double(),
        target.frames = col_double(),
        cue = col_factor(),
        task = col_factor(),
        target.type = col_factor(),
        rt.subj = col_double(),
        rt.obj = col_double(),
        even.digit = col_integer(),
        seed = col_double(),
        obj.resp = col_factor(),
        subject = col_factor()
    )
)
```

```
## Warning: The following named parsers don't match the column names: seed

## Warning: The following named parsers don't match the column names: seed

## Warning: The following named parsers don't match the column names: seed
```

```
# removing all the practice trials #ii
exp1 <- samples[samples$trial.type == "experiment", ]
exp1 <- mutate(exp1, # iii
    correct = as.logical(
        ifelse(substr(target.type, 1, 1) == obj.resp, 1, 0)
    )
)
```

iv. In the previous experiment, the number of target frames was fixed at 3, whereas this time it the target frames was not fixed (they were integer numbers from 1-6), and in the last experiment where target.contrast was not fixed but for this experiment target.contrast was fixed at 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.
    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.
    ii. the log-likelihood-function for logistic regression is: $l(p) = \sum_{i=1}^{N} [y_i \ln p + (1-y_i) \ln (1-p)$. Create a function that calculates the log-likelihood
    iii. apply both functions to the pooling model you just created. Make sure that the log-likelihood matches what is returned from the *logLik* function for the pooled model. Does the likelihood-function return a value that is surprising? Why is the log-likelihood preferable when working with computers with limited precision?
    iv. now show that the log-likelihood is a little off when applied to the partial pooling model - (the likelihood function is different for the multilevel function - see section 2.1 of https://www.researchgate.net/profile/Douglas-Bates/publication/2753537_Computational_Methods_for_Multilevel_Modelling/links/00b4953b4108d73427000000/Methods-for-Multilevel-Modelling.pdf (https://www.researchgate.net/profile/Douglas-Bates/publication/2753537_Computational_Methods_for_Multilevel_Modelling/links/00b4953b4108d73427000000/Methods-for-Multilevel-Modelling.pdf) if you are interested)
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.
    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 estimated group-level function with `xlim=c(0, 8)` that includes the estimated subject-specific functions.
    ii. also include in the results section whether the fit didn't look good for any of the subjects. If so, 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 %)
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
    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 behaviour reasonable?

Answers

Exercise 5, part 1 - KV

```
#creating a pooled model
m1 <- glm(correct ~ target.frames, data = exp1,
    family = binomial(link = "logit"))
summary(m1)
```

```
##
## Call:
## glm(formula = correct ~ target.frames, family = binomial(link = "logit"),
##     data = exp1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6452   0.2478   0.3546   0.7039   1.2621
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.92992    0.03485  -26.69   <2e-16 ***
## target.frames  0.73296    0.01219   60.11   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 26683  on 25043  degrees of freedom
## Residual deviance: 21730  on 25042  degrees of freedom
## AIC: 21734
##
## Number of Fisher Scoring iterations: 5
```

```
#creating a partial-pooled model
m2 <- glmer(correct ~ target.frames + (1 | subject),
    data = exp1, family = binomial(link = "logit"))
summary(m2)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ target.frames + (1 | subject)
##    Data: exp1
##
##      AIC      BIC   logLik deviance df.resid
##  21250.1  21274.4 -10622.0  21244.1    25041
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -7.7520  0.1436  0.2604  0.4816  2.0730
##
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.1549   0.3936
## Number of obs: 25044, groups:  subject, 29
##
## Fixed effects:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.95968    0.08125  -11.81   <2e-16 ***
## target.frames  0.75493    0.01251   60.37   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## target.frms -0.382
```

```
#5.i creating our own likelihood function
likelihood <- function(p, y) {
    prod((p^y) * (1 - p)^(1 - y))
}

#5.ii creating our own log-likelihood function
log_likelihood <- function(p, y) {
    sum(y * log(p) + (1 - y) * log(1 - p))
}

#5.iii Getting some likelihood and log-likelihood values and double-checking that our function spits out the righ
t values for m1
p <- fitted.values(m1)
likelihood(p, exp1$correct)
```

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

```
log_likelihood(p, exp1$correct)
```

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

```
(m1_ll <- logLik(m1))
```

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

```
# 5.iv Showing that the log-likelihood is slightly different when using _LogLik_ in comparison to our own functio
n for m2
p <- fitted.values(m2)
likelihood(p, exp1$correct)
```

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

```
log_likelihood(p, exp1$correct)
```

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

```
(m2_ll <- logLik(m2))
```

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

5.iii a likelihood value of 0 is returned for both m1 and m2, which is due to the lack of precision of the computer. This is why it is preferable to use the log-likelihood function instead.

5.iv The log-likelihood value returned from our function is -10565.53 whereas the value for the *LogLik* function is 10622.03, which can probably be explained by the fact that our function does not take multilevel modelling into considaration.

Exercise 5, part 2 - LR

```
#creating the null-model
m0 <- glm(correct ~ 1, data = exp1, family = binomial(link = "logit"))


#adding subject-level intercepts
m3 <- glmer(correct ~ 1 + (1 | subject),
    data = exp1, family = binomial(link = "logit"))

#adding subject-level intercepts and slopes
m4 <- glmer(correct ~ target.frames + (target.frames | subject),
    data = exp1, family = binomial(link = "logit"))


#checking the log-likelihood of the different models
(m0_ll <- logLik(m0))
```

```
## 'log Lik.' -13341.54 (df=1)
```

```
(m1_ll <- logLik(m1))
```

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

```
(m2_ll <- logLik(m2))
```

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

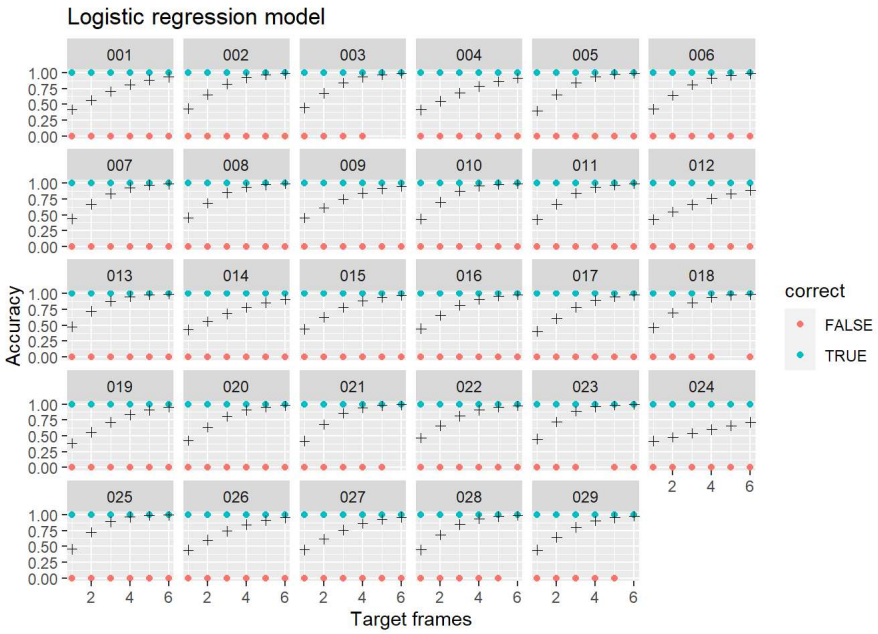
```
(m3_ll <- logLik(m3))
```

```
## 'log Lik.' -13157.55 (df=2)
```

```
(m4_ll <- logLik(m4))
```

```
## 'log Lik.' -10448.83 (df=5)
```

```
#creating some great plots to investigate the data
ggplot(exp1, aes(target.frames, as.numeric(correct), color = correct)) +
    geom_point() +
    geom_point(aes(target.frames, fitted.values(m4)),
        shape = 3, color = "black", inherit.aes = FALSE) +
    facet_wrap(~subject) +
    labs(
        x = "Target frames",
        y = "Accuracy",
        title = "Logistic regression model")
```




```
#zooming in on subject 024
(subj24_accuracy <- exp1 %>%
    filter(subject == "024") %>%
    summarise(accuracy = sum(correct) / n()))
```

```
## # A tibble: 1 x 1
##   accuracy
##      <dbl>
## 1    0.568
```

2.i+ii We have run binomial regression models, in which we tested the relationship between the variable *correct* predicted by *target.frames*. Different models using fixed and and random effects were created for comparison. The quality of the models Was compared by using the log-likelihood function.

Out of the 5 models, we choose m4 as our best model, as it has the highest LogLik value of -10676.02, with 5 degrees of freedom. After observing the plots, we have identified subject number 024 as having a bad fit. As the fit looks more linear than sigmoid, and has values all along the axis x = 0. Judging from this graph we predict that their performance is different to the other participants possibly having a more equal ratio of correct and incorrect, as they are the only subject that had a graph that resulted in looking more linear. We then took subject 024 and took the number of correct divided by trials to retrieve their accuracy score and compare it to chance, 50%. They have a accuracy percentage of 56.9%, which is only slightly higher than chance, which could explain why their graph looks poor.

Exercise 5, part 3 - VK

```
#Creating some models
m5 <- glmer(correct ~ target.frames + pas + (target.frames | subject),
    data = exp1, family = binomial(link = "logit"))
(m5_ll <- logLik(m5))
```

```
## 'log Lik.' -9931.828 (df=8)
```

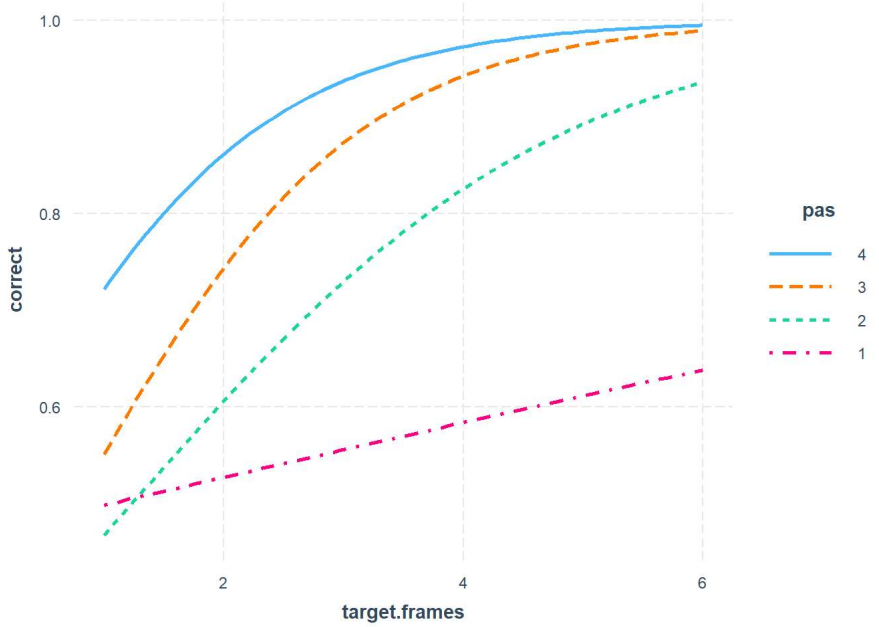
```
m6 <- glmer(correct ~ target.frames + pas +
    (pas * target.frames) + (target.frames | subject),
    data = exp1, family = binomial(link = "logit"), control = glmerControl(optimizer = "bobyqa"))
(m6_ll <- logLik(m6))
```

```
## 'log Lik.' -9742.039 (df=11)
```

```
summary(m6)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula:
## correct ~ target.frames + pas + (pas * target.frames) + (target.frames |
##     subject)
##    Data: exp1
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  19506.1  19595.5  -9742.0  19484.1    25033
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -19.0101   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)          0.08009    0.24320   0.329  0.74193
## target.frames        0.87404    0.06792  12.869  < 2e-16 ***
## pas3                -0.74022    0.27065  -2.735  0.00624 **
## pas2                -0.77311    0.25133  -3.076  0.00210 **
## pas1                -0.20172    0.24613  -0.820  0.41246
## target.frames:pas3 -0.01055    0.07335  -0.144  0.88565
## target.frames:pas2 -0.31206    0.06850  -4.555 5.23e-06 ***
## target.frames:pas1 -0.75924    0.06749 -11.250  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) trgt.f pas3   pas2   pas1   trg.:3 trg.:2
## target.frms -0.896
## pas3        -0.869  0.759
## pas2        -0.945  0.826  0.847
## pas1        -0.966  0.842  0.859  0.936
## trgt.frms:3  0.775 -0.776 -0.926 -0.761 -0.769
## trgt.frms:2  0.841 -0.841 -0.762 -0.921 -0.839  0.790
## trgt.frms:1  0.853 -0.850 -0.767 -0.838 -0.916  0.791  0.870
```

```
interactions::interact_plot(model = m6, pred = "target.frames", modx = "pas")
```



The log-likelihood of m4, m5 and m6 are these:

m4_ll = -10676.02, df=(5) m5_11 = -9931.828, df=(8) m6_ll = -9742.039, df=(11)

Thereby justifying the choice of model 6 as our preferred model

5.3 ii The chosen model, m6, is a binomial model in which the variable *correct* is predicted by *target.frames*, *PAS* and their interaction.

The estimate of *target.frames* shows that an increase in this variable “generally” increases accuracy - however, let’s look at the intercations_

We use this interaction between PAS and target duration because the lesser amount of frames (target duration) the less confident participants are (PAS).

Together, PAS with time duration (target.frames), the correctness gets worse as target.frames:PAS decreases (from PAS_4 to PAS_1). It can be read from the model summary that *target.frames* influences correctness less with lower PAS level (e.g. target.frames:PAS_1 = -0.75924 etc.)

The intercept of 0.08009, does sound strange as one might incorrectly expect it to be around 50%, but naturally this is because it is not possible to show a figure at a duration of 0 frames <3 The intercept of 0.08 is thus only meaningful within the model but not in reality.

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

Answers

Exercise 6, part 1 - EH

```
#creating the desired model
m7 <- glmer(correct ~ pas * target.frames +
    (target.frames | subject), data = exp1,
    family = binomial(link = "logit"),
    control = glmerControl(optimizer = "bobyqa"))
#getting that summary
summary(m7)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ pas * target.frames + (target.frames | subject)
##    Data: exp1
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  19506.1  19595.5  -9742.0  19484.1    25033
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -19.0102   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)         0.08010    0.24575   0.326  0.74446
## pas3               -0.74025    0.27355  -2.706  0.00681 **
## pas2               -0.77313    0.25370  -3.047  0.00231 **
## pas1               -0.20174    0.24871  -0.811  0.41729
## target.frames       0.87404    0.06850  12.760  < 2e-16 ***
## pas3:target.frames -0.01054    0.07403  -0.142  0.88678
## pas2:target.frames -0.31205    0.06904  -4.520 6.19e-06 ***
## pas1:target.frames -0.75924    0.06809 -11.151  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) pas3   pas2   pas1   trgt.f ps3:t. ps2:t.
## pas3        -0.871
## pas2        -0.946  0.850
## pas1        -0.966  0.862  0.937
## target.frms -0.898  0.763  0.829  0.845
## ps3:trgt.fr  0.780 -0.927 -0.766 -0.773 -0.780
## ps2:trgt.fr  0.843 -0.766 -0.923 -0.841 -0.844  0.794
## ps1:trgt.fr  0.856 -0.771 -0.841 -0.917 -0.853  0.794  0.872
```

6.1.i we see that the interaction-effect between *pas* and *target.frames* is clearly higher for PAS_2 vs PAS_1, thus indicating that accuracy increases faster with objective evidence for PAS_2 than for PAS_1.

Exercise 6, part 2 - KV

```
#trying out glht on m7
glht(m7)
```

```
##
##   General Linear Hypotheses
##
## Linear Hypotheses:
##                          Estimate
## (Intercept) == 0         0.08010
## pas3 == 0               -0.74025
## pas2 == 0               -0.77313
## pas1 == 0               -0.20174
## target.frames == 0       0.87404
## pas3:target.frames == 0 -0.01054
## pas2:target.frames == 0 -0.31205
## pas1:target.frames == 0 -0.75924
```

Snippet for 6.2.i+ii+iii

```
## testing whether PAS 2 is different from PAS 1
contrast.vector <- matrix(c(0, 0, -1, 1, 0, 0, 0, 0), nrow = 1)
gh <- glht(m7, contrast.vector)
print(summary(gh))
## as another example, we could also test whether there is a difference in
## intercepts between PAS 2 and PAS 3
contrast.vector <- matrix(c(0, -1, 1, 0, 0, 0, 0, 0), nrow = 1)
gh <- glht(m7, contrast.vector)
print(summary(gh))

## PAS 4 and 3
contrast.vector <- matrix(c(-1, 1, 0, 0, 0, 0, 0, 0), nrow = 1)
gh <- glht(m7, contrast.vector)
print(summary(gh))
```

Exercise 6, part 3 - LR

```
#creating the contrast_vector
K <- rbind(
    c(0, 0, -1, 1),
    c(-1, 1, 0, 0)
)

gh <- glht(m7, mcp(pas = K))
print(summary(gh))
```

6.3.i - as seen in the output there is a greater difference between PAS 2 and 1 than between 4 and 3.

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

```
sigfit <- function(a, b, c, d, x) {
    a + (b - a) / (1 + exp((c - x) / d))
}

RSS <- function(dataset, par) {
    x <- dataset$x
    y <- dataset$y
    y.hat <- sigfit(par[1], par[2], par[3], par[4], x)
    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)
    ii. Plot the fits for the PAS ratings on a single plot (for subject 7) `xlim=c(0, 8)`
    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)`
    iv. Comment on the differences between the fits - mention some advantages and disadvantages of each way
2. Finally, estimate the parameters for all subjects and each of their four PAS ratings. Then plot the estimated function at the group-level by taking the mean for each of the four parameters, *a*, *b*, *c* and *d* across subjects. A function should be estimated for each PAS-rating (it should look somewhat similar to Fig. 3 from the article: https://doi.org/10.1016/j.concog.2019.03.007 (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.

Answers

Exercise 7, part 1 - VK

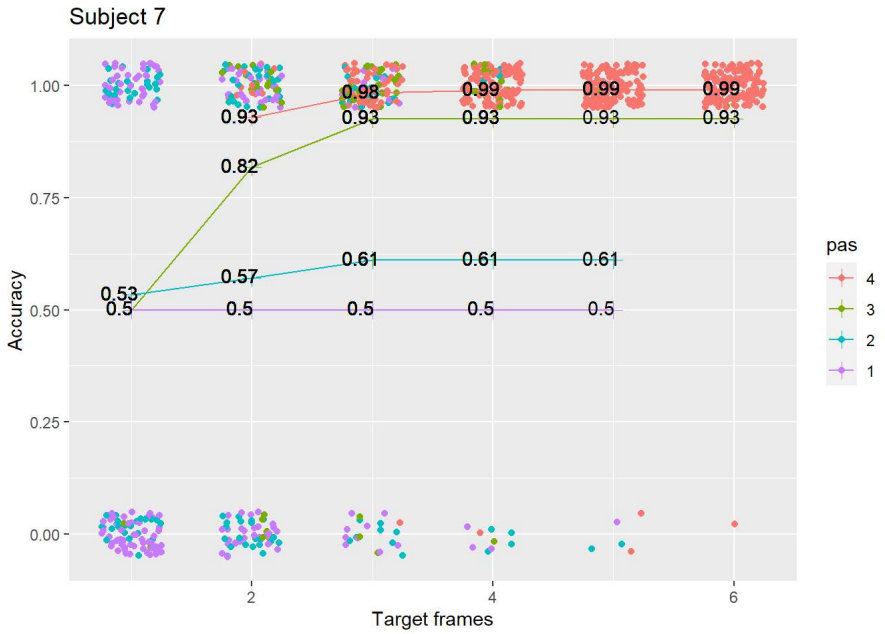
```r
#filtering out subject 7
subj7 <- exp1[exp1$subject == "007", ]

#choosing our desired variables
subj7xy <- data.frame(
    x = subj7$target.frames,
    y = subj7$correct,
    pas = subj7$pas)


#Setting up the function and using it on the four pas_levels


#PAS 1
s7p1 <- optim(
    par = c(0.5, 1, 1, 1), # we set a to be 0.5 as we expect chance level to be the appropriate minimum accuracy
    fn = RSS,
    data = subj7xy[subj7xy$pas == "1", ],
    method = "L-BFGS-B",
    lower = c(0.5, 0.5, -Inf, -Inf),
    upper = c(1, 1, Inf, Inf)
)
p1fit <- sigfit(
    a = s7p1$par[1],
    b = s7p1$par[2],
    c = s7p1$par[3],
    d = s7p1$par[4],
    x = subj7xy[subj7xy$pas == "1", ]$x
)

#PAS 2
s7p2 <- optim(
    par = c(0.5, 1, 1, 1),
    fn = RSS,
    data = subj7xy[subj7xy$pas == "2", ],
    method = "L-BFGS-B",
    lower = c(0.5, 0.5, -Inf, -Inf),
    upper = c(1, 1, Inf, Inf)
)

p2fit <- sigfit(
    a = s7p2$par[1],
    b = s7p2$par[2],
    c = s7p2$par[3],
    d = s7p2$par[4],
    x = subj7xy[subj7xy$pas == "2", ]$x
)

#PAS 3
s7p3 <- optim(
    par = c(0.5, 1, 1, 1),
    fn = RSS,
    data = subj7xy[subj7xy$pas == "3", ],
    method = "L-BFGS-B",
    lower = c(0.5, 0.5, -Inf, -Inf),
    upper = c(1, 1, Inf, Inf)
)

p3fit <- sigfit(
    a = s7p3$par[1],
    b = s7p3$par[2],
    c = s7p3$par[3],
    d = s7p3$par[4],
    x = subj7xy[subj7xy$pas == "3", ]$x
)

#PAS 4
s7p4 <- optim(
    par = c(0.5, 1, 1, 1),
    fn = RSS,
    data = subj7xy[subj7xy$pas == "4", ],
    method = "L-BFGS-B",
    lower = c(0.5, 0.5, -Inf, -Inf),
    upper = c(1, 1, Inf, Inf)
)

p4fit <- sigfit(
    a = s7p4$par[1],
    b = s7p4$par[2],
    c = s7p4$par[3],
    d = s7p4$par[4],
    x = subj7xy[subj7xy$pas == "4", ]$x
)


length(subj7xy[subj7xy$pas == "1", ]$x)
```

```
## [1] 183
```

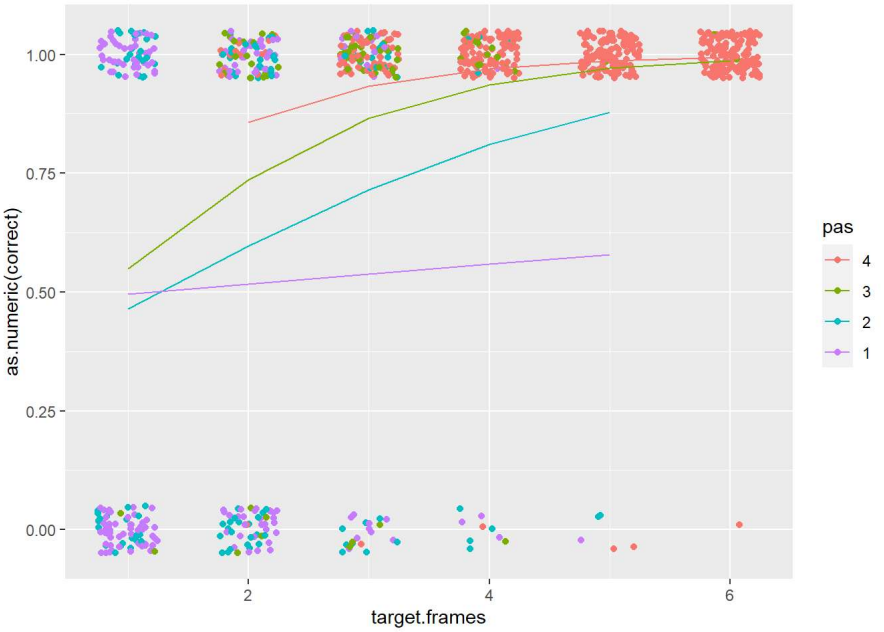
```
length(p1fit)
```

```
## [1] 183
```

```
subj7xy$yhat <- NA
subj7xy[subj7xy$pas == "1", ]$yhat <- p1fit
subj7xy[subj7xy$pas == "2", ]$yhat <- p2fit
subj7xy[subj7xy$pas == "3", ]$yhat <- p3fit
subj7xy[subj7xy$pas == "4", ]$yhat <- p4fit

#plotting the fits of the sigmoid
ggplot(aes(x = x, y = as.numeric(y), color = pas), data = subj7xy) +
    geom_jitter(width = 0.25, height = 0.05) +
    geom_line(aes(y = yhat)) +
    geom_point(aes(y = yhat), shape = 3, size = 3) +
    geom_text(aes(y = yhat, label = round(yhat, 2)),
        color = "black", nudge_x = -0.1, nudge_y = 0.005
    ) +
    labs(x = "Target frames", y = "Accuracy", title = "Subject 7")
```



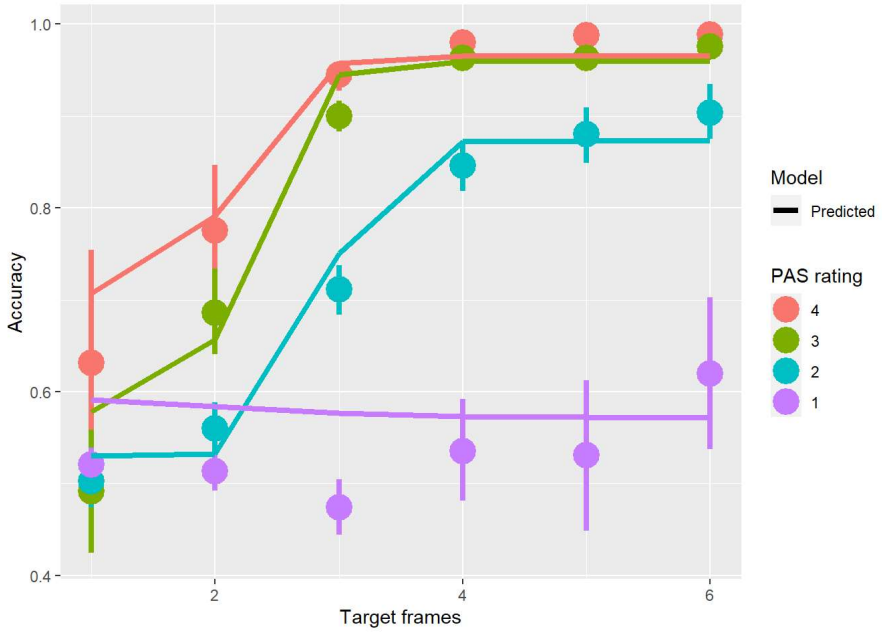

```
s7m7fit <- predict(m7, newdata = subj7, type = "response")

#plotting m7
ggplot(aes(x = target.frames, y = as.numeric(correct),color = pas),
    data = subj7) +
    geom_jitter(width = 0.25, height = 0.05) +
    geom_line(aes(y = s7m7fit))
```



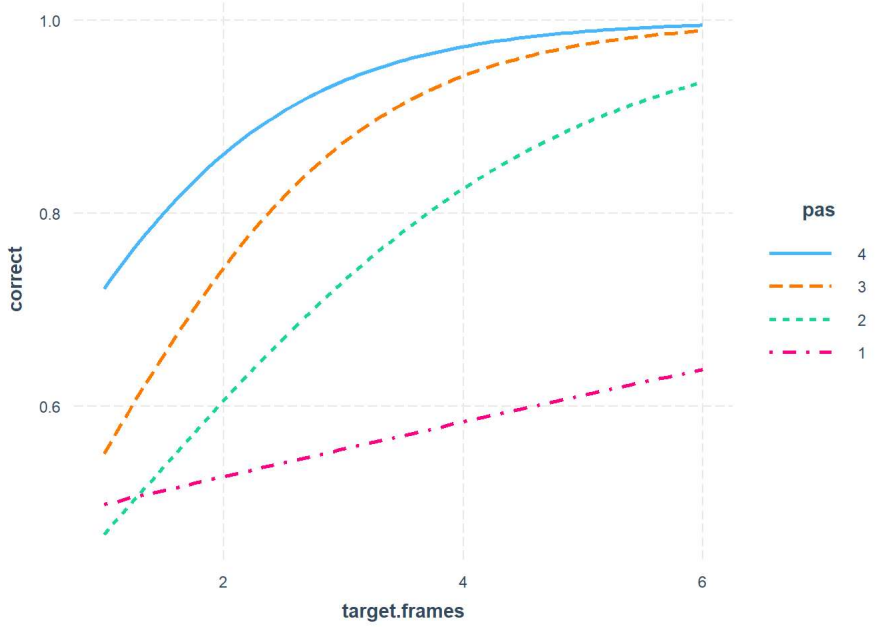
7.1.iv As we plotted the predicted values of the fits, the intercepts at *target.frames*=0 is not shown, though this would have been a crucial difference between the plots.The visible difference between the plots must primarily stem from the fact that we manually set the lowest possible value to 0.5 in our optim function. The advantage of manually creating the function is that we can control for the fact that below chance level accuracy is highly unlikely and thus avoid an uninterpretable intercept, however we might be prone to bias as we set the parameters ourselves. On the other hand, using the binomial model might prone to less bias as we do not decide that any parameters, but might return (as in this case) values which are uninterpretable (as the intercept in this).It should however be noticed that the plot of m7 is based upon all the subject data whereas the “optim”-plot is only based on subject 7, thus making them not entirely comparable.

Exercise 7, part 2 - EH

```r
sigmodel_pas <- function(pas_lev, subjx) {
    dat <- data.frame(
        x = subjx[subjx$pas == pas_lev, ]$target.frames,
        y = subjx[subjx$pas == pas_lev, ]$correct)
    subjxpx_opt <- optim(
        par = c(0.5, 1, 1, 1),
        fn = RSS,
        data = dat,
        method = "L-BFGS-B",
        lower = c(0.5, 0.5, -Inf, -Inf),
        upper = c(1, 1, Inf, Inf)
    )

    list(
        subjxpx_opt$par[1],
        subjxpx_opt$par[2],
        subjxpx_opt$par[3],
        subjxpx_opt$par[4]
    )
}

sigmodel <- function(x, dat, paslevs) {
    subjx <- dat[dat$subject == x, ]
    sapply(paslevs, sigmodel_pas, subjx = subjx,
        USE.NAMES = FALSE, simplify = FALSE)
}

fit_sigmodel <- function(pas, dat, params) {
    sigfit(
        a = params[, pas][1],
        b = params[, pas][2],
        c = params[, pas][3],
        d = params[, pas][4],
        x = dat[dat$pas == pas, ]$target.frames
    )
}

pars <- c("a", "b", "c", "d")
subjects <- levels(exp1$subject)
paslevs <- levels(exp1$pas)
N <- length(subjects)

sigmodel_params <- array(
    unlist(sapply(
        subjects,
        sigmodel,
        dat = exp1,
        paslevs = paslevs,
        USE.NAMES = FALSE,
        simplify = FALSE
    )),
    dim = c(4, 4, N), dimnames = list(pars, paslevs, subjects)
)

inter_subj_par_means_by_pas <- rowMeans(sigmodel_params, dims = 2)



sig_fitted <- sapply(
    paslevs,
    fit_sigmodel,
    dat = exp1,
    params = inter_subj_par_means_by_pas
)

exp1$y_hat <- NA
exp1[exp1$pas == 1, ]$y_hat <- sig_fitted$`1`
exp1[exp1$pas == 2, ]$y_hat <- sig_fitted$`2`
exp1[exp1$pas == 3, ]$y_hat <- sig_fitted$`3`
exp1[exp1$pas == 4, ]$y_hat <- sig_fitted$`4`

ggplot(exp1, aes(target.frames, as.numeric(correct), color = pas)) +
    stat_summary(fun.data = "mean_cl_boot", size = 1.5) +
    geom_line(
        aes(x = target.frames, y = y_hat, linetype = "Predicted"),
        size = 1.5
    ) +
    labs(
        x = "Target frames",
        y = "Accuracy",
        color = "PAS rating",
        linetype = "Model"
    )
```




```
interactions::interact_plot(model = m6, pred = "target.frames", modx = "pas")
```




```
summary(m6)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula:
## correct ~ target.frames + pas + (pas * target.frames) + (target.frames |
##     subject)
##    Data: exp1
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  19506.1  19595.5  -9742.0  19484.1    25033
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -19.0101   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)         0.08009    0.24320   0.329  0.74193
## target.frames       0.87404    0.06792  12.869  < 2e-16 ***
## pas3               -0.74022    0.27065  -2.735  0.00624 **
## pas2               -0.77311    0.25133  -3.076  0.00210 **
## pas1               -0.20172    0.24613  -0.820  0.41246
## target.frames:pas3 -0.01055    0.07335  -0.144  0.88565
## target.frames:pas2 -0.31206    0.06850  -4.555 5.23e-06 ***
## target.frames:pas1 -0.75924    0.06749 -11.250  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) trgt.f pas3   pas2   pas1   trg.:3 trg.:2
## target.frms -0.896
## pas3        -0.869  0.759
## pas2        -0.945  0.826  0.847
## pas1        -0.966  0.842  0.859  0.936
## trgt.frms:3  0.775 -0.776 -0.926 -0.761 -0.769
## trgt.frms:2  0.841 -0.841 -0.762 -0.921 -0.839  0.790
## trgt.frms:1  0.853 -0.850 -0.767 -0.838 -0.916  0.791  0.870
```

7.2.i Of course the plot for the optim function looks a bit wonky (as it is based on the predicted values), but the shape of the slope can still be assessed. A crucial difference between the plots is the slope of PAS 1, which is positive for m6 and negative for the plot of the optim function. As in the previous exercise, the advantage of the optim function is that you can control the parameters resulting in avoidance of below chance level estimates - but as we decide the parameters it might be prone to bias. For m6, we use the inbuilt binomial function making us less prone to bias, but giving us estimates which are uniterpretable.