Portfolio Assignment 1, Methods 3, 2021, autumn semester

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

Student numbers

Emma Rose Hahn:202004249 Viara Krasteva: 202005673 Kristian Nøhr Villebro: 202005289 Luke Ring: 202009983

Portfolio Assignment 1: Using mixed effects modelling to model hierarchical data

In this assignment we will be investigating the *politeness* dataset of Winter and Grawunder (2012) and apply basic methods of multilevel modelling.

```
#this is the dataset we will be exploring in this exercise
politeness <- read.csv('politeness.csv') ## read in data
```

Exercise 1 - describing the dataset and making some initial plots

1. Describe the dataset, such that someone who happened upon this dataset could understand the variables and what they contain

Exercise 1, part 1– (EH)

The politeness dataset contains the data obtained from the study of Korean formal and informal speech (https://doi.org/10.1016/j.wocn.2012.08.006) which investigated the fundamental frequency of male and female participants' speech in a variety of formal and informal scenarios.

The following table describes the variables in the dataset:

Variable	Description
`subject`	participant ID
`gender`	participant's gender
`scenario`	the experimental scenario from 1 to 7 such as "asking a favour"
`attitude`	either 'inf' for informal stimuli or 'pol' for formal stimuli
`total_duration`	duration of participant's response in seconds
`f0mn`	mean fundamental frequency (f0) of the participant's speech
`hiss_count`	number of times the participants made a noisy breath intake

Remark: The `gender`, `scenario` and `attitude` variables should be encoded as factors as they show a categorical function withing this dataset . In addition, these variables have non-unique values across participants, and are not ordered.

```
#Encoding some of the variables as factors (gender, attitude, and scenario)

politeness$attitude <- as.factor(politeness$attitude)
politeness$gender <- as.factor(politeness$gender)
politeness$scenario <- as.factor(politeness$scenario)
```

Exercise 1, part 2–(EH)

2. Create a new data frame that just contains the subject *F1* and run two linear models; one that expresses *f0mn* as dependent on *scenario* as an integer; and one that expresses *f0mn* as dependent on *scenario* encoded as a factor
   i. Include the model matrices, X from the General Linear Model, for these two models in your report and describe the different interpretations of *scenario* that these entail
   ii. Which coding of *scenario*, as a factor or not, is more fitting?

```
# Create a subset dataframe for subject F1 only
pf1 <- politeness[politeness$subject == "F1", ]
pf1
```

```
##    subject gender scenario attitude total_duration  f0mn hiss_count
## 1       F1      F        1      pol         18.392 214.6          2
## 2       F1      F        1      inf         13.551 210.9          0
## 3       F1      F        2      pol          5.217 284.7          0
## 4       F1      F        2      inf          4.247 265.6          0
## 5       F1      F        3      pol          6.791 210.6          0
## 6       F1      F        3      inf          4.126 285.6          0
## 7       F1      F        4      pol          6.244 251.5          1
## 8       F1      F        4      inf          3.245 281.5          0
## 9       F1      F        5      pol          5.625 229.6          1
## 10      F1      F        5      inf          3.950 250.5          0
## 11      F1      F        6      pol         28.508 181.1          1
## 12      F1      F        6      inf         55.159 229.3          0
## 13      F1      F        7      inf         60.309 219.8          2
## 14      F1      F        7      pol         40.825 175.8          0
```

```
# make model predicting f0mn by scenario (integer)
m1<- lm(f0mn ~ as.integer(scenario), data = pf1)

# get model matrix
mm1 <- model.matrix(m1)

# make model predicting f0mn by scenario (factor)
m2 <- lm(f0mn ~ as.factor(scenario), data = pf1)

# get model matrix
mm2 <- model.matrix(m2)
```

```
Here is the model using "scenario" encoded as an integer
```

```
summary(m1)
```

```
##
## Call:
## lm(formula = f0mn ~ as.integer(scenario), data = pf1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -44.836 -36.807   6.686  20.918  46.421
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           262.621     20.616  12.738 2.48e-08 ***
## as.integer(scenario)   -6.886      4.610  -1.494    0.161
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 34.5 on 12 degrees of freedom
## Multiple R-squared:  0.1568, Adjusted R-squared:  0.0865
## F-statistic: 2.231 on 1 and 12 DF,  p-value: 0.1611
```

```
mm1
```

```
##    (Intercept) as.integer(scenario)
## 1            1                    1
## 2            1                    1
## 3            1                    2
## 4            1                    2
## 5            1                    3
## 6            1                    3
## 7            1                    4
## 8            1                    4
## 9            1                    5
## 10           1                    5
## 11           1                    6
## 12           1                    6
## 13           1                    7
## 14           1                    7
## attr(,"assign")
## [1] 0 1
```

And here is the model using “scenario” encoded as a factor

```
summary(m2)
```

```
##
## Call:
## lm(formula = f0mn ~ as.factor(scenario), data = pf1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -37.50 -13.86   0.00  13.86  37.50
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            212.75      20.35  10.453  1.6e-05 ***
## as.factor(scenario)2    62.40      28.78   2.168   0.0668 .
## as.factor(scenario)3    35.35      28.78   1.228   0.2591
## as.factor(scenario)4    53.75      28.78   1.867   0.1041
## as.factor(scenario)5    27.30      28.78   0.948   0.3745
## as.factor(scenario)6    -7.55      28.78  -0.262   0.8006
## as.factor(scenario)7   -14.95      28.78  -0.519   0.6195
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.78 on 7 degrees of freedom
## Multiple R-squared:  0.6576, Adjusted R-squared:  0.364
## F-statistic:  2.24 on 6 and 7 DF,  p-value: 0.1576
```

```
mm2
```

```
##    (Intercept) as.factor(scenario)2 as.factor(scenario)3 as.factor(scenario)4
## 1            1                    0                    0                    0
## 2            1                    0                    0                    0
## 3            1                    1                    0                    0
## 4            1                    1                    0                    0
## 5            1                    0                    1                    0
## 6            1                    0                    1                    0
## 7            1                    0                    0                    1
## 8            1                    0                    0                    1
## 9            1                    0                    0                    0
## 10           1                    0                    0                    0
## 11           1                    0                    0                    0
## 12           1                    0                    0                    0
## 13           1                    0                    0                    0
## 14           1                    0                    0                    0
##    as.factor(scenario)5 as.factor(scenario)6 as.factor(scenario)7
## 1                     0                    0                    0
## 2                     0                    0                    0
## 3                     0                    0                    0
## 4                     0                    0                    0
## 5                     0                    0                    0
## 6                     0                    0                    0
## 7                     0                    0                    0
## 8                     0                    0                    0
## 9                     1                    0                    0
## 10                    1                    0                    0
## 11                    0                    1                    0
## 12                    0                    1                    0
## 13                    0                    0                    1
## 14                    0                    0                    1
## attr(,"assign")
## [1] 0 1 1 1 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$`as.factor(scenario)`
## [1] "contr.treatment"
```

Conclusion: The above output shows the difference in model matrices between scenario encoded as an integer and factor. The integer version treats scenario as a continuous variable, whereas the factorized version creates a regression line per scenario.

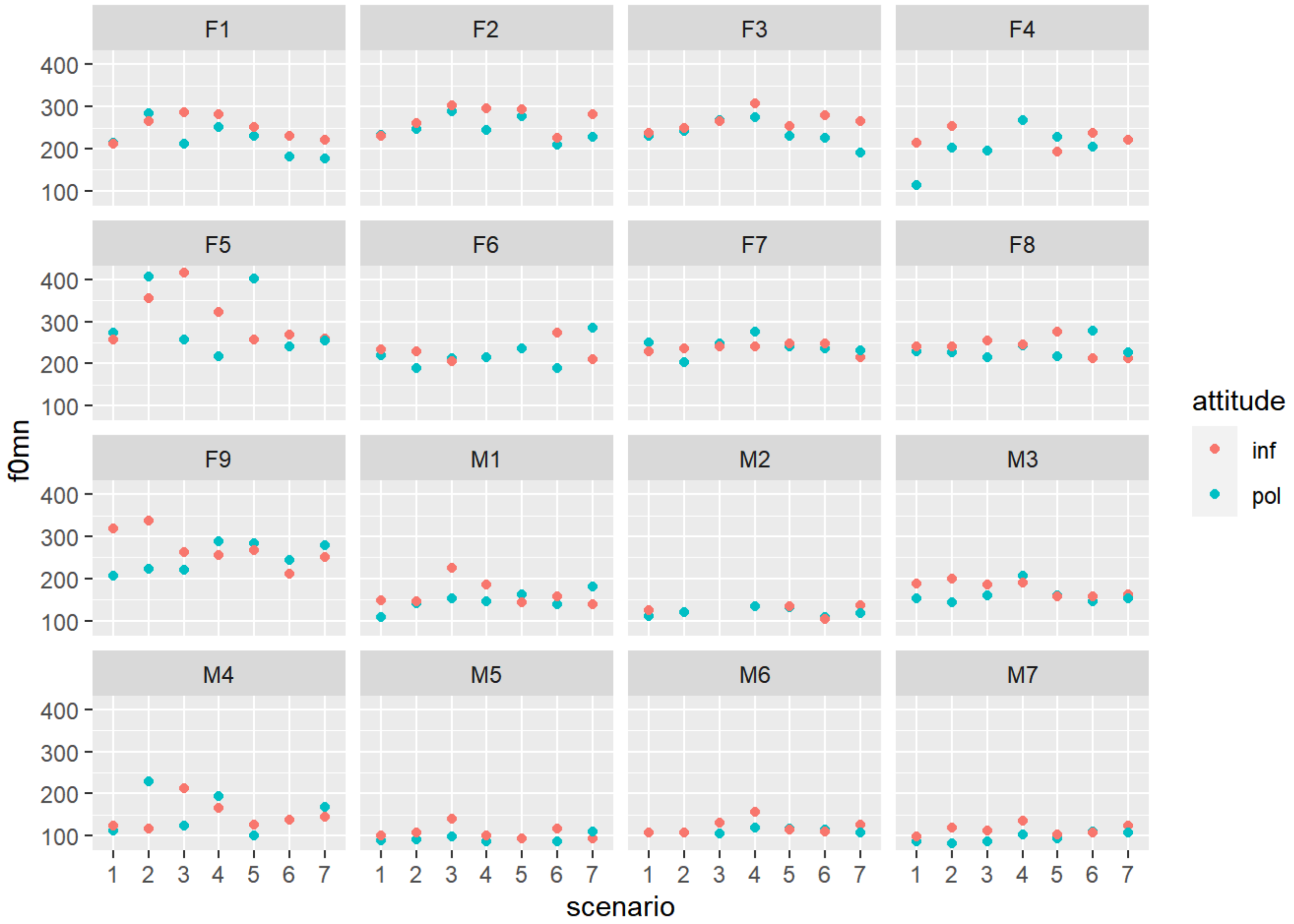
For this dataset, scenario should be a factor, since the scenarios are not a continuous variable and depending on the prescribed scenario, the participants may have a different f0 (mean fundamental frequency of speech), and we are interested in following the trajectory of f0 across scenarios not as a variable that consistently decreases or increases, but a separate regression line showing the changes in f0 between the 7 different scenarios. And in order, to be able to see that crucial difference we need to consider the ‘scenario’ variable as a factor when we run a model predicting f0 across scenarios.

Exercise 1, part 3 – (EH)

3. Make a plot that includes a subplot for each subject that has *scenario* on the x-axis and *f0mn* on the y-axis and where points are colour coded according to *attitude*
    i. Describe the differences between subjects

```
politeness %>% ggplot(aes(scenario, f0mn, color = attitude)) +
    geom_point() +
    facet_wrap(vars(subject))
```

```
## Warning: Removed 12 rows containing missing values (geom_point).
```



…

We can visually observe that there are baseline differences between the male and female subjects’ mean fundamental frequency of speech, where the males’ f0 is consistently lower, across scenario and attitude. Between the different scenarios there is variability in the f0 values for both male and female subjects depending on both the scenario type and the attitude (informal or formal). There is a consistent tendency across scenario type and gender for the mean fundamental frequency of speech to be slightly higher when the attitude is informal as opposed to formal. This visual information inerpreted from this plot is consistent with the results of Winter and Grawunder (2012)

Exercise 2 - comparison of models

```
mixed.model <- lmer(formula=..., data=...)
example.formula <- formula(dep.variable ~ first.level.variable + (1 | second.level.va
riable))
```

Exercise 2, Part 1 – (VK)

1. Build four different models and do some comparisons

```r
# the single level model
m3 <- lm(formula = f0mn ~ gender, data = politeness)

# a two-level model where each scenario has a unique intercept
m4 <- lmer(formula = f0mn ~ gender + (1 | scenario), data = politeness)
# a two-level model that has models subject as intercept
m5 <- lmer(formula = f0mn ~ gender + (1 | subject), data = politeness)

# a two-level model that incorporate intercepts for both subject and scenario
m6 <- lmer(formula = f0mn ~ gender +
            (1 | subject) + (1 | scenario), data = politeness)

#comparing AIC and Deviance values for all the models
AIC(m3)
```

```
## [1] 2163.971
```

```r
AIC(m4)
```

```
## [1] 2152.314
```

```r
AIC(m5)
```

```
## [1] 2099.626
```

```r
AIC(m6)
```

```
## [1] 2092.482
```

```r
deviance(m3)
```

```
## [1] 327033.6
```

```r
deviance(m4)
```

```
## Warning in deviance.merMod(m4): deviance() is deprecated for REML fits;
## use REMLcrit for the REML criterion or deviance(.,REML=FALSE) for deviance
## calculated at the REML fit
```

```
## [1] 2144.314
```

```r
deviance(m5)
```

```
## Warning in deviance.merMod(m5): deviance() is deprecated for REML fits;
## use REMLcrit for the REML criterion or deviance(.,REML=FALSE) for deviance
## calculated at the REML fit
```

```
## [1] 2091.626
```

```
deviance(m6)
```

```
## Warning in deviance.merMod(m6): deviance() is deprecated for REML fits;
## use REMLcrit for the REML criterion or deviance(.,REML=FALSE) for deviance
## calculated at the REML fit
```

```
## [1] 2082.482
```

```
anova(m4, m5, m6)
```

```
## refitting model(s) with ML (instead of REML)
```

```
## Data: politeness
## Models:
## m4: f0mn ~ gender + (1 | scenario)
## m5: f0mn ~ gender + (1 | subject)
## m6: f0mn ~ gender + (1 | subject) + (1 | scenario)
##    npar    AIC    BIC  logLik deviance   Chisq Df Pr(>Chisq)
## m4    4 2162.3 2175.7 -1077.1   2154.3
## m5    4 2112.1 2125.5 -1052.0   2104.1 50.2095  0
## m6    5 2105.2 2122.0 -1047.6   2095.2  8.8725  1   0.002895 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
piecewiseSEM::rsquared(c(m4, m5, m6))
```

```
##   Response   family     link method  Marginal Conditional
## 1     f0mn gaussian identity   none 0.6779555   0.6967788
## 2     f0mn gaussian identity   none 0.6681651   0.7899229
## 3     f0mn gaussian identity   none 0.6677206   0.8077964
```

The single level model performs the worst and this makes sense as we do not expect all participants to have the same f0 as their voices have naturally occurring differences (not just ones predicted by gender). There are differences that might be explained by the scenario or individual subject (as we observed in the plot above), however, neither of those are taken into account when using a single level model.

So then using two-level models that explain variance by either taking scenario or subject as random intercepts, is definitely an improvement to help explain more of the scenario/attitude based differences, not just the gender differences.

Consequently, Of the three multi-level models,it is model m6, which includes random intercepts for both subject and scenario,that has the most explained variance with for the entire model $R^2 \approx 0.81$ or 81%.

Additionally, we see that model m6 has the lowest AIC and deviance.

Exercise 2, part 2 and 3 – (LR)

2. Why is our single-level model bad?
    i. create a new data frame that has three variables, *subject*, *gender* and *f0mn*, where *f0mn* is the average of all responses of each subject, i.e. averaging across *attitude* and_scenario_
    ii. build a single-level model that models *f0mn* as dependent on *gender* using this new dataset
    iii. make Quantile-Quantile plots, comparing theoretical quantiles to the sample quantiles) using `qqnorm` and `qqline` for the new single-level model and compare it to the old single-level model (from 1).i). Which model's residuals (ϵ) fulfil the assumptions of the General Linear Model better?)
    iv. Also make a quantile-quantile plot for the residuals of the multilevel model with two intercepts. Does it look alright?
3. Plotting the two-intercepts model
    i. Create a plot for each subject, (similar to part 3 in Exercise 1), this time also indicating the fitted value for each of the subjects for each for the scenarios (hint use `fixef` to get the "grand effects" for each gender and `ranef` to get the subject- and scenario-specific effects)

```
# scenario x f0mn y, attitude = color
ff <- fixef(m6)
ff
```

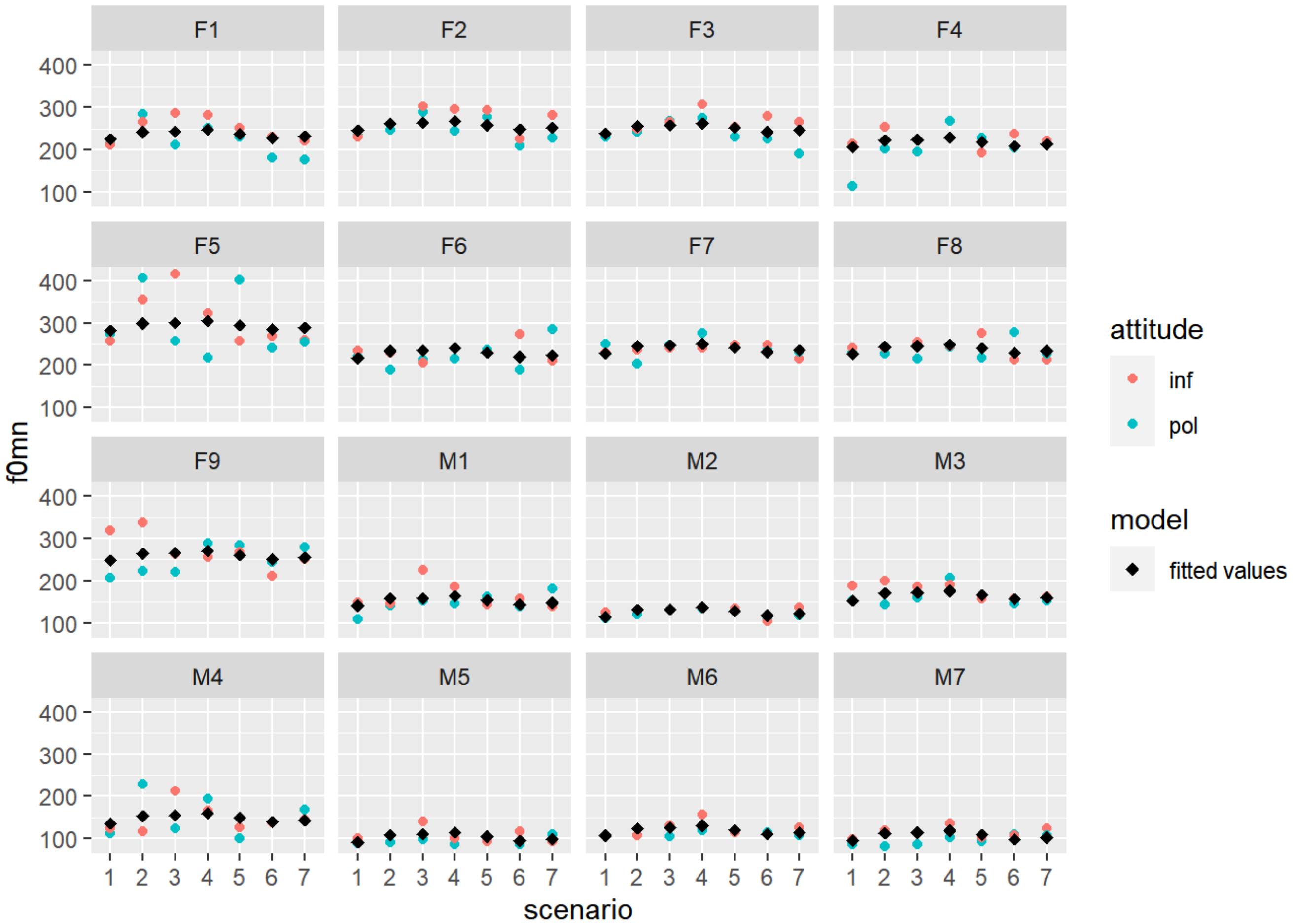
```
## (Intercept)     genderM
##    246.7650   -115.1746
```

```
rf <- ranef(m6)
rf <- as.data.frame(rf)
rf
```

```
##      grpvar        term grp    condval   condsd
## 1   subject (Intercept)  F1 -10.490356 8.280794
## 2   subject (Intercept)  F2  10.251809 8.280794
## 3   subject (Intercept)  F3   3.795129 8.280794
## 4   subject (Intercept)  F4 -29.495270 9.095403
## 5   subject (Intercept)  F5  47.093999 8.280794
## 6   subject (Intercept)  F6 -18.396273 8.794359
## 7   subject (Intercept)  F7  -6.976691 8.280794
## 8   subject (Intercept)  F8  -8.521934 8.280794
## 9   subject (Intercept)  F9  12.739587 8.280794
## 10  subject (Intercept)  M1  21.052117 8.280794
## 11  subject (Intercept)  M2  -5.462358 9.453009
## 12  subject (Intercept)  M3  33.561535 8.280794
## 13  subject (Intercept)  M4  16.093337 8.524543
## 14  subject (Intercept)  M5 -28.267430 8.280794
## 15  subject (Intercept)  M6 -12.640202 8.794541
## 16  subject (Intercept)  M7 -24.336998 8.280794
## 17 scenario (Intercept)   1 -11.595496 5.488728
## 18 scenario (Intercept)   2   5.321218 5.532205
## 19 scenario (Intercept)   3   6.795658 5.586194
## 20 scenario (Intercept)   4  11.348815 5.578013
## 21 scenario (Intercept)   5   1.411037 5.488705
## 22 scenario (Intercept)   6  -8.622136 5.489258
## 23 scenario (Intercept)   7  -4.659096 5.488058
```

```
politeness$effect_gender <- 0.0
politeness[politeness$gender == "F", ]$effect_gender <- ff[1]
politeness[politeness$gender == "M", ]$effect_gender <- ff[1] + ff[2]
politeness$intercept_subject <- left_join(politeness, rf, by = c("subject" = "grp"),
 copy = TRUE, keep = FALSE)$condval
politeness$intercept_scenario <- left_join(politeness, rf, by = c("scenario" = "grp"
), copy = TRUE, keep = FALSE)$condval
politeness$predicted <- politeness$effect_gender + politeness$intercept_subject + pol
iteness$intercept_scenario
politeness %>% ggplot(aes(scenario, f0mn, color = attitude)) +
    geom_point() +
    geom_point(aes(y = predicted, shape = "fitted values"), color = "black", size = 2
) +
    scale_shape_manual(name = "model", values = c(18)) +
    facet_wrap(vars(subject))
```

```
## Warning: Removed 12 rows containing missing values (geom_point).
```




```
deviance(m3)
```

```
## [1] 327033.6
```

```
deviance(m4, REML = FALSE)
```

```
## [1] 2154.33
```

```
deviance(m5, REML = FALSE)
```

```
## [1] 2104.175
```

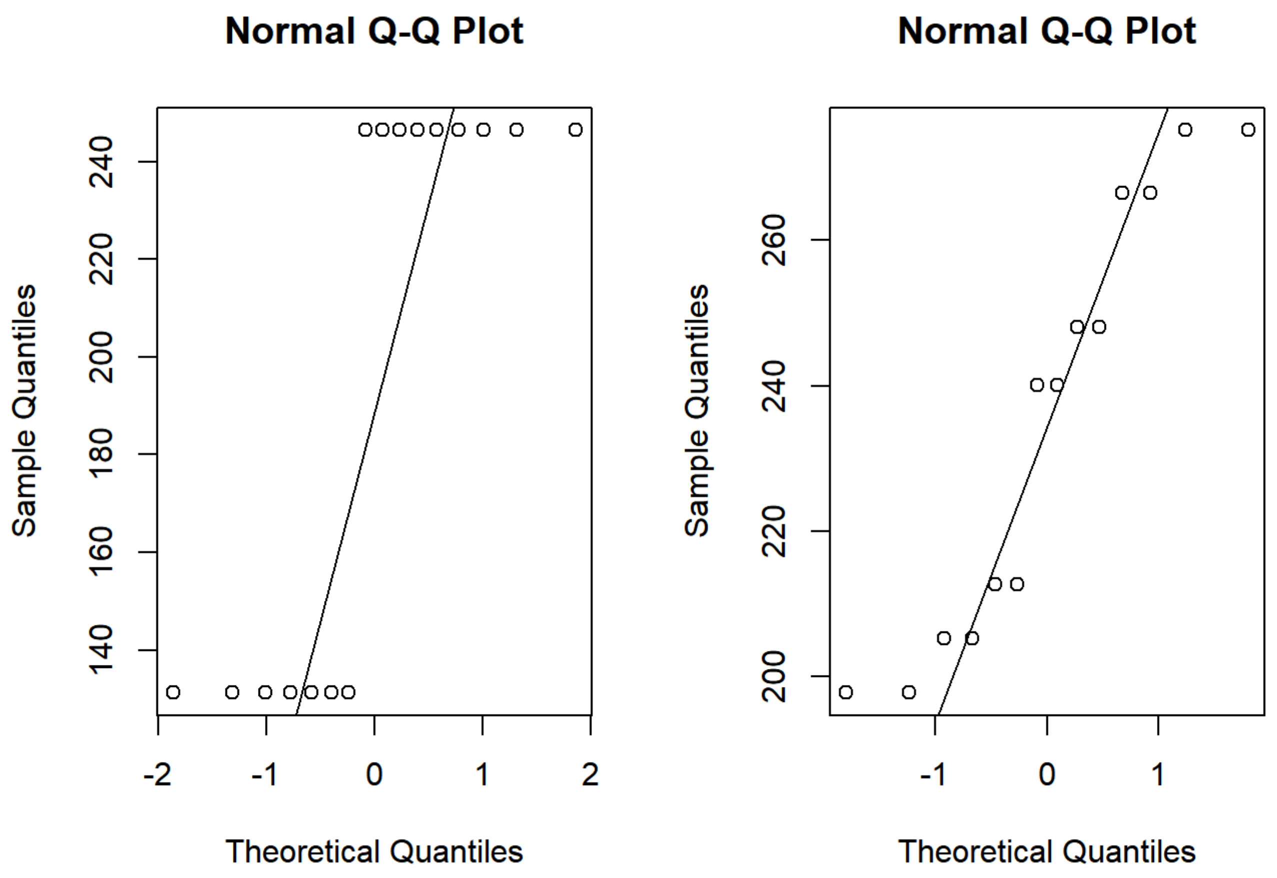
```
deviance(m6, REML = FALSE)
```

```
## [1] 2095.279
```

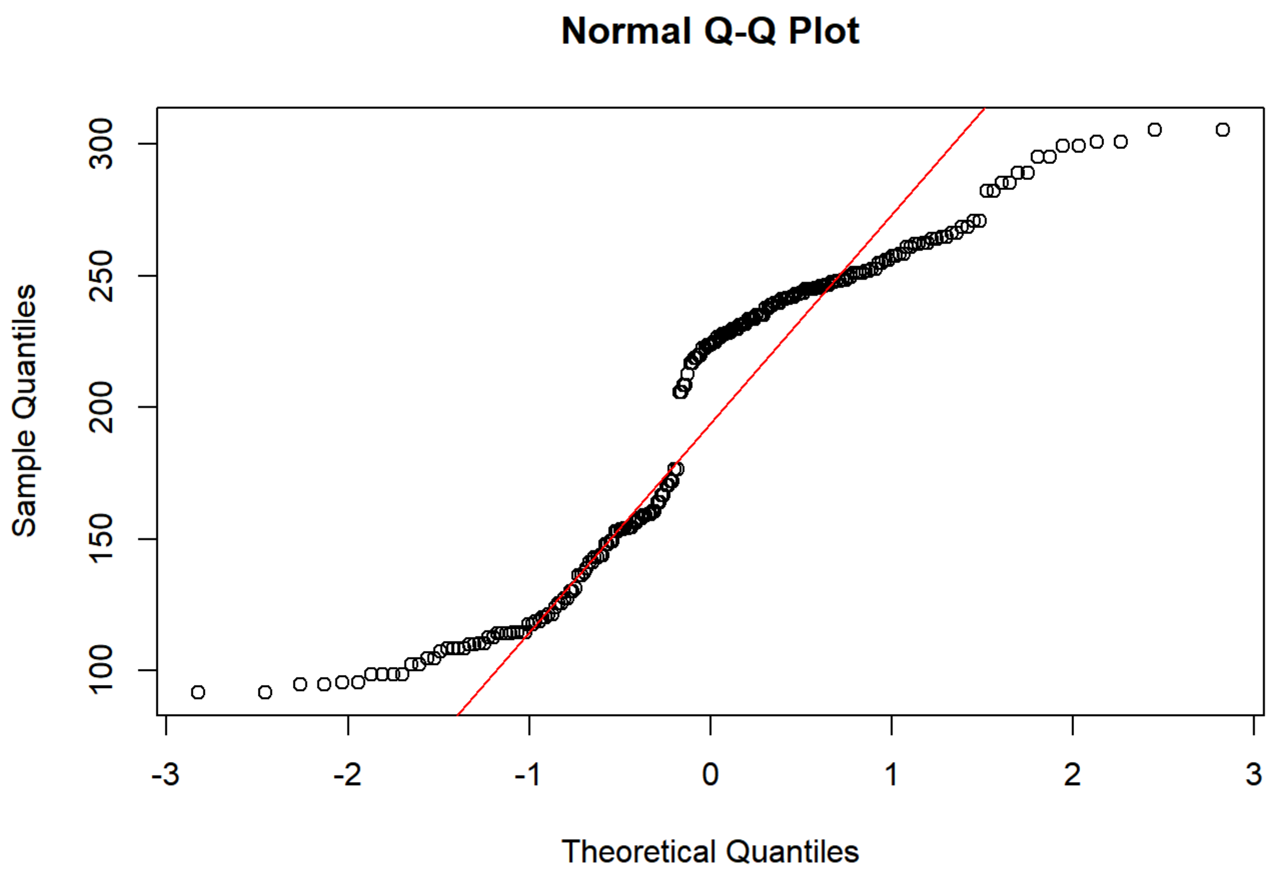
```
politeness_aggregated <- politeness[!is.na(politeness$f0mn), ] %>% group_by(subject)
%>% summarize(subject = subject[1], gender = gender[1], f0mn = mean(f0mn))
politeness_aggregated
```

```
## # A tibble: 16 x 3
##    subject gender  f0mn
##    <chr>   <fct>  <dbl>
##  1 F1      F       235.
##  2 F2      F       258.
##  3 F3      F       251.
##  4 F4      F       212.
##  5 F5      F       299.
##  6 F6      F       225.
##  7 F7      F       239.
##  8 F8      F       237.
##  9 F9      F       261.
## 10 M1      M       155.
## 11 M2      M       122.
## 12 M3      M       169.
## 13 M4      M       150.
## 14 M5      M       100.
## 15 M6      M       118.
## 16 M7      M       104.
```

```
m7 <- lm(f0mn ~ gender, data = politeness_aggregated)
par(mfrow=c(1,2))
qqnorm(fitted.values(m7))
qqline(fitted.values(m7))
qqnorm(fitted.values(m2))
qqline(fitted.values(m2))
```




```
par(mfrow=c(1,1))
qqnorm(fitted.values(m6))
qqline(fitted.values(m6), col = "red")
```



Assessing the QQ-plots of the single-level models it seems that the aggregated model m7's residuals are worse off than those of model m2. The residuals of model m2 are better dispersed along the line - however it still doesn't look fantastic. The QQ-plot of the multilevel model m6 looks better than any of the single level ones, with data points closer to the line and more evenly dispersed on both sides of the line.

Looking at the plot for the observed and the fitted values it looks as if the model m6 performs reasonably as well.

Exercise 3 - now with attitude

Exercise 3, part 1 –(VK)

1. Carry on with the model with the two unique intercepts fitted (*scenario* and *subject*) but now build a new model that has *attitude* as a main effect besides *gender*. After create a separate model that besides the main effects of *attitude* and *gender* also include their interaction

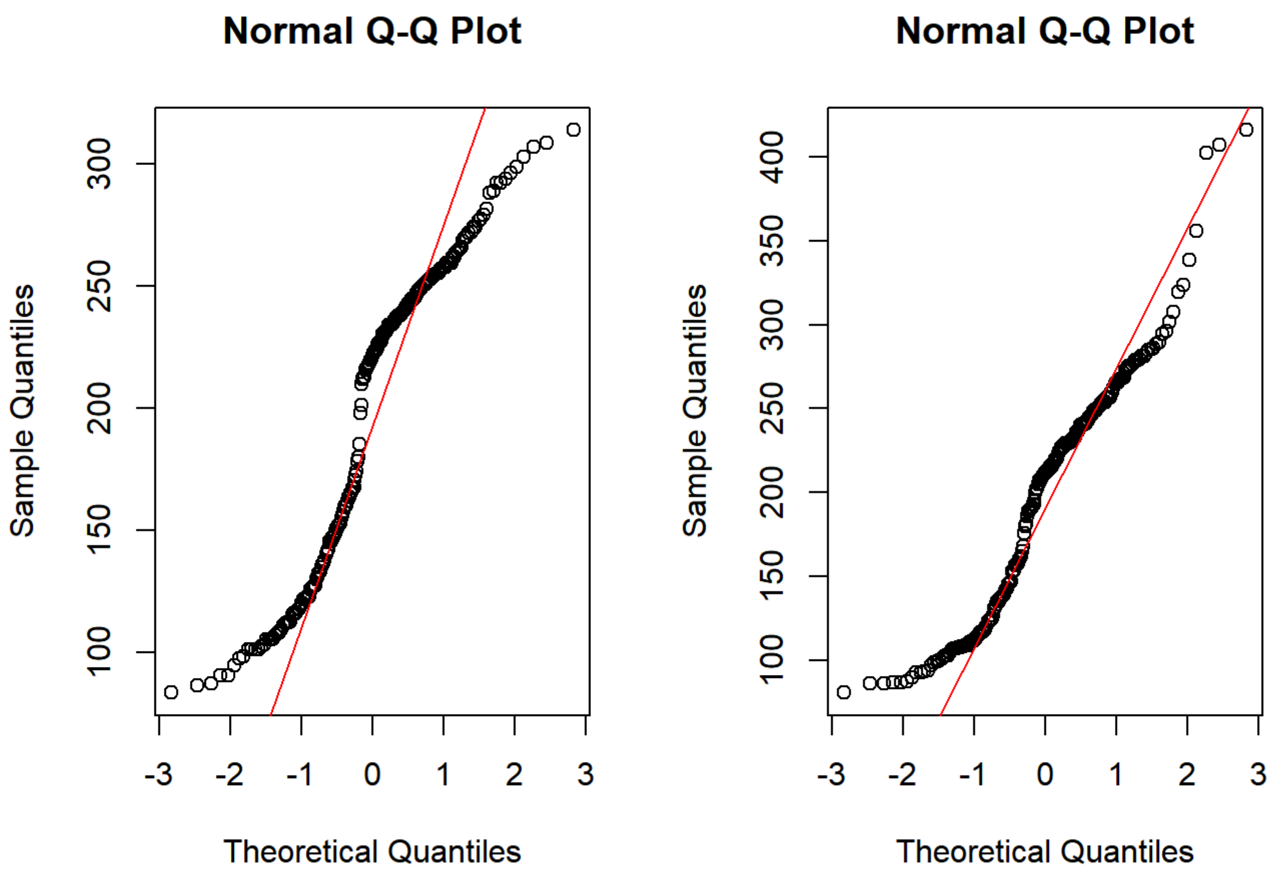
```
#making a model with two unicue intercepts and having attitude and gender as main eff
ects
m8 <- lmer(formula = f0mn ~ gender + attitude + (1 | subject) + (1 | scenario), data
 = politeness)
summary(m8)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: f0mn ~ gender + attitude + (1 | subject) + (1 | scenario)
##    Data: politeness
##
## REML criterion at convergence: 2065.1
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.8511 -0.6081 -0.0602  0.4329  3.8745
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  subject  (Intercept) 585.6    24.20
##  scenario (Intercept) 106.7    10.33
##  Residual             882.7    29.71
## Number of obs: 212, groups:  subject, 16; scenario, 7
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  254.398      9.597  26.507
## genderM     -115.437     12.881  -8.962
## attitudepol  -14.819      4.096  -3.618
##
## Correlation of Fixed Effects:
##             (Intr) gendrM
## genderM     -0.587
## attitudepol -0.220  0.006
```

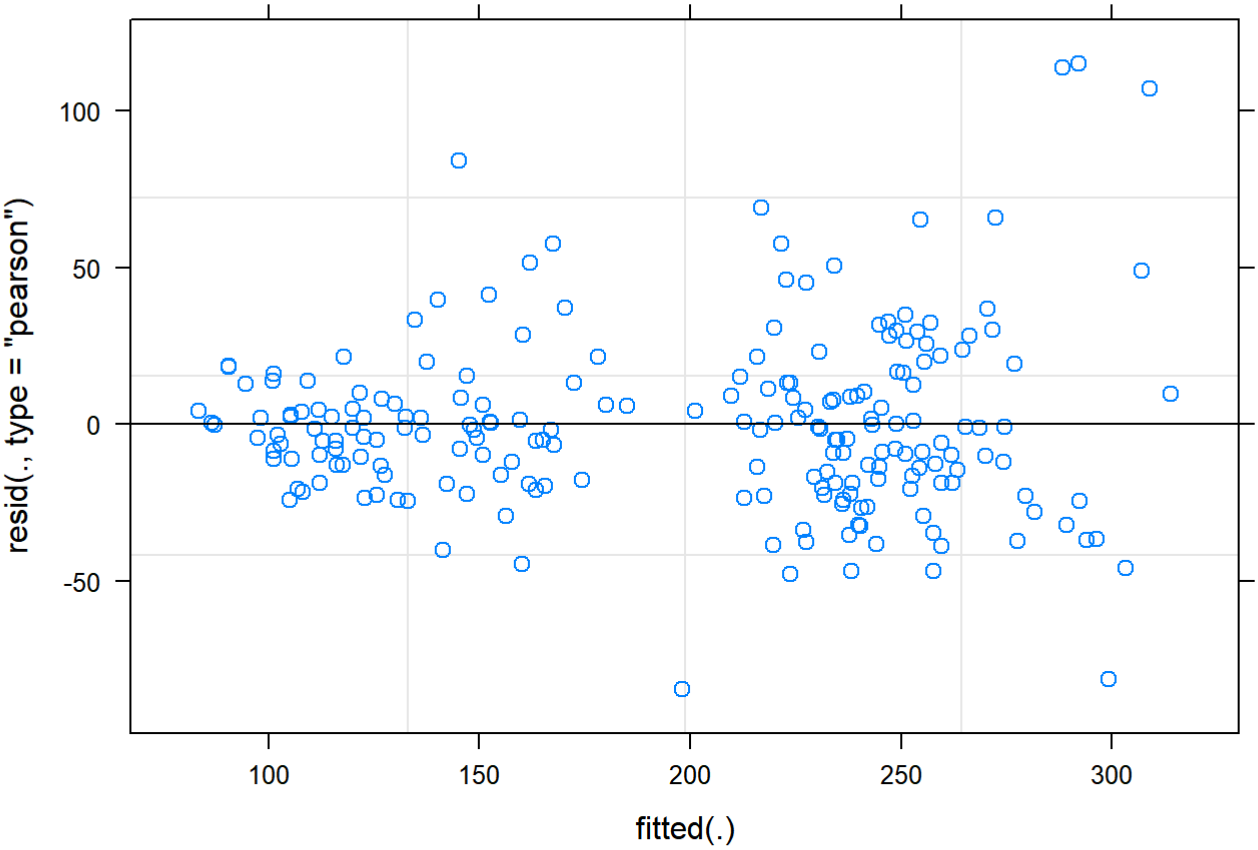
```
fitted.values(m8)
```

```
##         1         2         3         4         5         6         7         8 
## 216.47929 231.29853 234.27048 249.08973 236.09087 250.91011 241.21606 256.03530 
##         9        10        11        12        13        14        15        16 
## 230.41934 245.23858 219.58594 234.40518 238.54395 223.72471 237.34031 252.15956 
##        17        18        19        20        21        22        23        24 
## 255.13151 269.95075 256.95189 271.77114 262.07708 276.89632 251.28036 266.09960 
##        25        26        27        28        29        30        31        32 
## 240.44696 255.26621 259.40498 244.58573 230.84663 245.66588 248.63783 263.45707 
##        33        34        35        36        37        38        39        40 
## 250.45822 265.27746 255.58340 270.40264 244.78668 259.60593 233.95328 248.77253 
##        41        42        43        44        45        46        47        49 
## 252.91130 238.09206 198.00782 212.82706 215.79901 230.61826 217.61940 222.74458 
##        51        52        53        54        55        57        58        59 
## 211.94787 226.76711 201.11447 215.93371 220.07248 274.39362 289.21286 292.18481 
##        60        61        62        63        64        65        66        67 
## 307.00406 294.00520 308.82444 299.13038 313.94963 288.33367 303.15291 277.50027 
##        68        69        70        71        72        73        74        75 
## 292.31951 296.45828 281.63904 209.70329 224.52253 227.49448 242.31372 229.31487 
##        76        77        79        81        82        83        84        85 
## 244.13411 234.44005 223.64333 212.80994 227.62918 231.76795 216.94871 220.01309 
##        86        87        88        89        90        91        92        93 
## 234.83233 237.80428 252.62353 239.62467 254.44391 244.74985 259.56910 233.95314 
##        94        95        96        97        98        99       100       101 
## 248.77238 223.11974 237.93898 242.07775 227.25851 218.45899 233.27823 236.25019 
##       102       103       104       105       106       107       108       109 
## 251.06943 238.07057 252.88982 243.19576 258.01500 232.39904 247.21828 221.56564 
##       110       111       112       113       114       115       116       117 
## 236.38488 240.52366 225.70441 239.84235 254.66159 257.63354 272.45278 259.45393 
##       118       119       120       121       122       123       124       125 
## 274.27317 264.57911 279.39835 253.78240 268.60164 242.94900 257.76824 261.90701 
##       126       127       128       129       130       131       132       133 
## 247.08777 133.00242 147.82166 150.79361 165.61285 152.61400 167.43324 157.73918 
##       134       135       136       137       138       139       140       141 
## 172.55843 146.94247 161.76171 136.10907 150.92831 155.06708 140.24784 107.79414 
##       142       143       147       149       150       151       152       153 
## 122.61338 125.58533 132.53090 121.73419 136.55343 110.90079 125.72003 129.85880 
##       154       155       156       157       158       159       160       161 
## 115.03956 145.58352 160.40276 163.37471 178.19395 165.19510 180.01434 170.32028 
##       162       163       164       165       166       167       168       169 
## 185.13952 159.52357 174.34281 148.69017 163.50941 167.64818 152.82894 127.46793 
##       170       171       172       173       174       175       176       177 
## 142.28717 145.25913 160.07837 147.07951 161.89876 152.20470 167.02394 141.40798 
##       178       180       181       182       183       184       185       186 
## 156.22722 145.39382 149.53260 134.71335  83.40025  98.21950 101.19145 116.01069 
##       187       188       189       190       191       192       193       194 
## 103.01184 117.83108 108.13702 122.95626  97.34030 112.15955  86.50690 101.32615 
##       195       196       198       200       201       202       203       204 
## 105.46492  90.64568 112.78128 130.57248 117.57362 132.39287 122.69881 137.51805 
##       205       206       207       208       209       210       211       212 
## 111.90209 126.72133 101.06869 115.88793 120.02671 105.20746  87.35321 102.17245 
##       213       214       215       216       217       218       219       220 
## 105.14440 119.96365 106.96479 121.78403 112.08998 126.90922 101.29326 116.11250 
##       221       222       223       224 
##  90.45986 105.27910 109.41787  94.59863
```

```
par(mfrow=c(1,2))
qqnorm(fitted.values(m8))
qqline(fitted.values(m8), col = "red")
qqnorm(politeness$f0mn)
qqline(politeness$f0mn, col = "red")
```




```
plot(m8)
```




```
#a model that additionally includes the interaction between the main effects (attitud
e and gender)
m9 <- lmer(formula = f0mn ~ gender + attitude + gender:attitude + (1 | subject) + (1
 | scenario), data = politeness)
summary(m9)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: f0mn ~ gender + attitude + gender:attitude + (1 | subject) +
##     (1 | scenario)
##    Data: politeness
##
## REML criterion at convergence: 2058.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.8120 -0.5884 -0.0645  0.4014  3.9100
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  subject  (Intercept) 584.4    24.17
##  scenario (Intercept) 106.4    10.32
##  Residual             885.5    29.76
## Number of obs: 212, groups:  subject, 16; scenario, 7
##
## Fixed effects:
##                      Estimate Std. Error t value
## (Intercept)           255.618      9.761  26.186
## genderM              -118.232     13.531  -8.738
## attitudepol           -17.192      5.423  -3.170
## genderM:attitudepol     5.544      8.284   0.669
##
## Correlation of Fixed Effects:
##             (Intr) gendrM atttdp
## genderM     -0.606
## attitudepol -0.286  0.206
## gndrM:tttdp  0.187 -0.309 -0.654
```

The model m9 can be read as following: The intercept for women/inf are $\approx 256Hz$, when we look at men with the same attitude their pitch drops by $\approx 118Hz$. Overall a polite attitude will result in a drop in pitch by $\approx 17Hz$, however for men it will only be $-17.2 + 5.5 \approx 11.6Hz$. Korean women's relative drop in pitch is therefore larger than male's in a polite situation according to this sample.

Exercise 3, part 2 –(KV)

2. Compare the three models (1. gender as a main effect; 2. gender and attitude as main effects; 3. gender and attitude as main effects and the interaction between them.

```
#comparing the models
anova(m6, m8, m9)
```

```
## refitting model(s) with ML (instead of REML)
```

```
## Data: politeness
## Models:
## m6: f0mn ~ gender + (1 | subject) + (1 | scenario)
## m8: f0mn ~ gender + attitude + (1 | subject) + (1 | scenario)
## m9: f0mn ~ gender + attitude + gender:attitude + (1 | subject) + (1 | scenario)
##    npar    AIC    BIC  logLik deviance   Chisq Df Pr(>Chisq)
## m6    5 2105.2 2122.0 -1047.6   2095.2
## m8    6 2094.5 2114.6 -1041.2   2082.5 12.6868  1  0.0003683 ***
## m9    7 2096.0 2119.5 -1041.0   2082.0  0.4551  1  0.4998998
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
piecewiseSEM::rsquared(c(m6, m8, m9))
```

```
##   Response   family     link method  Marginal Conditional
## 1     f0mn gaussian identity   none 0.6677206   0.8077964
## 2     f0mn gaussian identity   none 0.6782542   0.8196777
## 3     f0mn gaussian identity   none 0.6782490   0.8192531
```

Exercise 3, part 3 –(KV)

3. Choose the model that you think describe the data the best - and write a short report on the main findings based on this model.

```
summary(m8)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: f0mn ~ gender + attitude + (1 | subject) + (1 | scenario)
##    Data: politeness
##
## REML criterion at convergence: 2065.1
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.8511 -0.6081 -0.0602  0.4329  3.8745
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  subject  (Intercept) 585.6    24.20
##  scenario (Intercept) 106.7    10.33
##  Residual             882.7    29.71
## Number of obs: 212, groups:  subject, 16; scenario, 7
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  254.398      9.597  26.507
## genderM     -115.437     12.881  -8.962
## attitudepol  -14.819      4.096  -3.618
##
## Correlation of Fixed Effects:
##             (Intr) gendrM
## genderM     -0.587
## attitudepol -0.220  0.006
```

This dataset consists of the basic demographic information of 16 Korean participants, and their observed pitch in different situations that require either an informal or polite(formal) attitude.

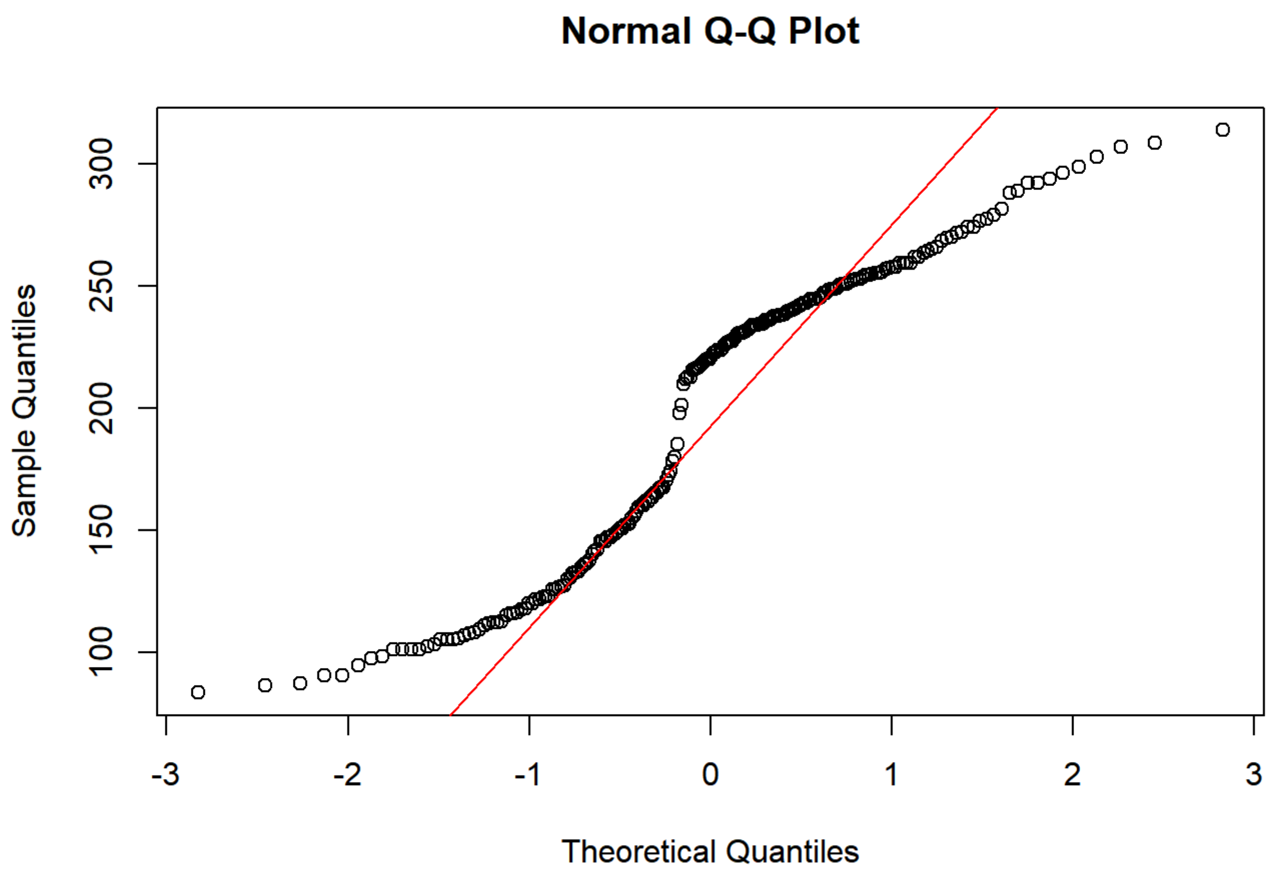
The model we chose as the best is m8. Consistent with the conclusions of the original oauthor's study, this model showed that women on average have a higher pitch than men BUT it also suggested a negative relationship between *attitude* and pitch with p-values<0.001. It would seem that both Korean men's and women's frequency of voice drops when having a polite attitude.

Subjects and scenarios should have different intercepts because it would be assumed they would all have different baselines, (and therefore need different intercepts to account for this). Different subjects will naturally already speak at a different pitch level, so separate intercepts allows us to account for these differences. The different scenarios may also need separate intercepts as certain scenarios may result in participants lowering or raising their pitch to meet the appropriate ambiance of the scenario. Once again, by including separate intercepts for scenarios then we should be accounting for these differences in our models.

Furthermore the output of the summary function shows that more variance is explained by the random effect of the subject than that of the scenario, further strengthening the choice of multilevel modelling.

And here's a QQ-plot of our chosen model

```
qqnorm(fitted.values(m8))
qqline(fitted.values(m8), col = "red")
```


Normal Q-Q Plot
Sample Quantiles
100
150
200
250
300
-3
-2
-1
0
1
2
3
Theoretical Quantiles

Portfolio Assignment 2 part 1, Methods 3, 2021, autumn semester

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

Student numbers

Emma Rose Hahn: 202004249 Viara Krasteva: 202005673 Kristian Nøhr Villebro: 202005289 Luke Ring: 202009983

Exercises and objectives

The objectives of the exercises of this assignment are:
1) Download and organise the data and model and plot staircase responses based on fits of logistic functions
2) Fit multilevel models for response times
3) Fit multilevel models for count data

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 assignment will be part of your final portfolio

Exercise 1

Go to https://osf.io/ecxsj/files/ (https://osf.io/ecxsj/files/) and download the files associated with Experiment 2 (there should be 29).
The data is associated with Experiment 2 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
2. Describe the data and construct extra variables from the existing variables
    i. add a variable to the data frame and call it *correct* (have it be a *logical* variable). Assign a 1 to each row where the subject indicated the correct answer and a 0 to each row where the subject indicated the incorrect answer (**Hint:** the variable *obj.resp* indicates whether the subject answered “even”, *e* or “odd”, *o*, and the variable *target_type* indicates what was actually presented.
    ii. describe what the following variables in the data frame contain, *trial.type*, *pas*, *trial*, *target.contrast*, *cue*, *task*, *target_type*, *rt.subj*, *rt.obj*, *obj.resp*,

subject and correct. (That means you can ignore the rest of the variables in your description). For each of them, indicate and argue for what `class` they should be classified into, e.g. *factor*, *numeric* etc.

iii. for the staircasing part **only**, create a plot for each subject where you plot the estimated function (on the *target.contrast* range from 0-1) based on the fitted values of a model (use `glm`) that models *correct* as dependent on *target.contrast*. These plots will be our *no-pooling* model. Comment on the fits - do we have enough data to plot the logistic functions?

iv. on top of those plots, add the estimated functions (on the *target.contrast* range from 0-1) for each subject based on partial pooling model (use `glmer` from the package `lme4`) where unique intercepts and slopes for *target.contrast* are modelled for each *subject*

v. in your own words, describe how the partial pooling model allows for a better fit for each subject

Answers

Exercise 1, part 1 - EH

```
# get list of all CSV files
temp <- list.files(path = "./experiment_2/",
    pattern = "*.csv", full.names = TRUE)

# load into single data frame
samples <- map_df(temp, read_csv, trim_ws = TRUE, na = c("", "NA"),
    col_types = cols(
        trial.type = col_factor(),
        pas = col_integer(),
        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()
    ))

rm(temp)
# peek data
head(samples)
```

```
## # A tibble: 6 x 17
##   trial.type   pas trial jitter.x jitter.y odd.digit target.contrast
##   <fct>      <int> <fct>    <dbl>    <dbl>     <int>           <dbl>
## 1 staircase      4 0      -0.343    0.449          9             1
## 2 staircase      4 1       0.0623   0.0291         9             1
## 3 staircase      4 2      -0.406    0.500          7             0.9
## 4 staircase      4 3      -0.362   -0.222          7             0.9
## 5 staircase      4 4       0.289    0.413          7             0.8
## 6 staircase      4 5       0.0824  -0.0934         7             0.8
## # ... with 10 more variables: target.frames <dbl>, cue <fct>, task <fct>,
## #   target.type <fct>, rt.subj <dbl>, rt.obj <dbl>, even.digit <int>,
## #   seed <dbl>, obj.resp <fct>, subject <fct>
```

```
# total number of samples
nrow(samples)
```

```
## [1] 18131
```

Exercise 1, part 2 - KV

```
# Add column to indicate if participant response was correct
samples <- mutate(samples,
    correct = as.logical(
        ifelse(substr(target.type, 1, 1) == obj.resp, 1, 0)
    )
)
```

The dataset contains the following variables:

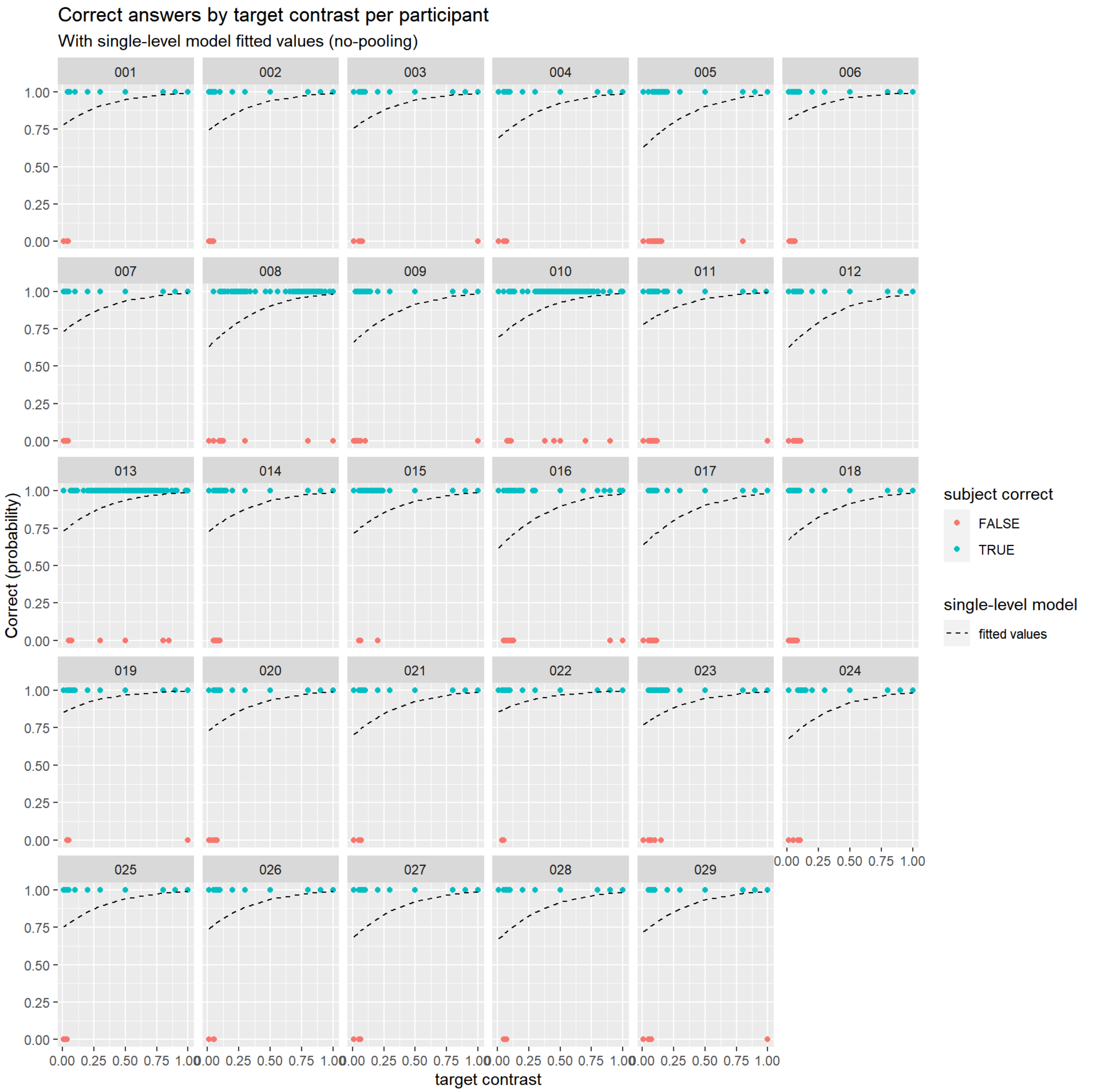
Variable	Description	Class
`trial.type`	either staircase(practice) or experiment	factor: categorical, reused
`pas`	any number from 1-4, indicating the reported experience on the Perceptual Awareness Scale	factor: categorical, not continuous
`trial`	trial number zero-indexed for the practice and experiment blocks	factor: categorical, not continuous
`target.contrast`	the grey-scale proportion of the target digit	double: numeric, continous variable from 0-1

Variable	Description	Class
`cue`	number code for pre-stimulus cue	factor: categorical, not continuous
`task`	how many numbers could be shown; has the levels quadruplet: (all numbers); pairs: (2 even and 2 odd numbers); singles: (1 even and 1 odd number)	factor: categorical, no continuous relationship
`target_type`	whether the target shown was the chosen even.digit or the chosen odd.digit	factor: categorical, no continuous relationship
`rt.subj`	reaction time (seconds) on the PAS response	double: continous variable with decimal places
`rt.obj`	reaction time (seconds) on the target digit	double: continous variable with decimal places
`obj.resp`	the key actually pressed e for even and o for odd	factor: categorical, no continuous relationship
`subject`	subject number	factor: categorical
`correct`	Whether answer was correct(1) or incorret (0)	logical, boolean either true or false

```
# only use staircase trial types.
samples_staircase <- samples %>% filter(trial.type == "staircase")
#making a no-pooling model
m1 <- glm(correct ~ target.contrast + subject,
    data = samples_staircase,
    family = binomial(link = "logit"))
# We should see an intercept per subject
summary(m1)
```

```
##
## Call:
## glm(formula = correct ~ target.contrast + subject, family = binomial(link = "logi
t"),
##     data = samples_staircase)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2225   0.1836   0.7262   0.7907   0.9772
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.237950   0.310228   3.990 6.59e-05 ***
## target.contrast  3.446129   0.439005   7.850 4.17e-15 ***
## subject002      -0.200038   0.396544  -0.504   0.6139
## subject003      -0.126797   0.367239  -0.345   0.7299
## subject004      -0.447583   0.341680  -1.310   0.1902
## subject005      -0.729034   0.342889  -2.126   0.0335 *
## subject006       0.204637   0.412265   0.496   0.6196
## subject007      -0.257607   0.397907  -0.647   0.5174
## subject008      -0.728765   0.345702  -2.108   0.0350 *
## subject009      -0.600476   0.339050  -1.771   0.0766 .
## subject010      -0.450820   0.353083  -1.277   0.2017
## subject011      -0.003327   0.353925  -0.009   0.9925
## subject012      -0.751795   0.339166  -2.217   0.0267 *
## subject013      -0.251208   0.354408  -0.709   0.4784
## subject014      -0.279840   0.345979  -0.809   0.4186
## subject015      -0.330359   0.349908  -0.944   0.3451
## subject016      -0.781364   0.340545  -2.294   0.0218 *
## subject017      -0.691345   0.339823  -2.034   0.0419 *
## subject018      -0.546979   0.340267  -1.607   0.1079
## subject019       0.502524   0.488717   1.028   0.3038
## subject020      -0.271168   0.356758  -0.760   0.4472
## subject021      -0.399562   0.346301  -1.154   0.2486
## subject022       0.523698   0.488385   1.072   0.2836
## subject023      -0.065056   0.359321  -0.181   0.8563
## subject024      -0.531046   0.341990  -1.553   0.1205
## subject025      -0.160067   0.426526  -0.375   0.7075
## subject026      -0.222237   0.374445  -0.594   0.5528
## subject027      -0.487602   0.342189  -1.425   0.1542
## subject028      -0.547571   0.339972  -1.611   0.1073
## subject029      -0.335918   0.355318  -0.945   0.3445
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6127.2  on 5602  degrees of freedom
## Residual deviance: 5936.1  on 5573  degrees of freedom
## AIC: 5996.1
##
## Number of Fisher Scoring iterations: 6
```

```
# plot the fitted values per subject
samples_staircase %>%
    ggplot(aes(target.contrast, as.integer(correct), color = correct)) +
    geom_point() +
    geom_line(aes(target.contrast, fitted(m1),
        linetype = "fitted values"), inherit.aes = FALSE) +
    scale_linetype_manual(name = "single-level model", values = c("dashed")) +
    facet_wrap(~subject) +
    labs(title = "Correct answers by target contrast per participant",
        subtitle = "With single-level model fitted values (no-pooling)",
        color = "subject correct") +
    ylab("Correct (probability)") +
    xlab("target contrast")
```



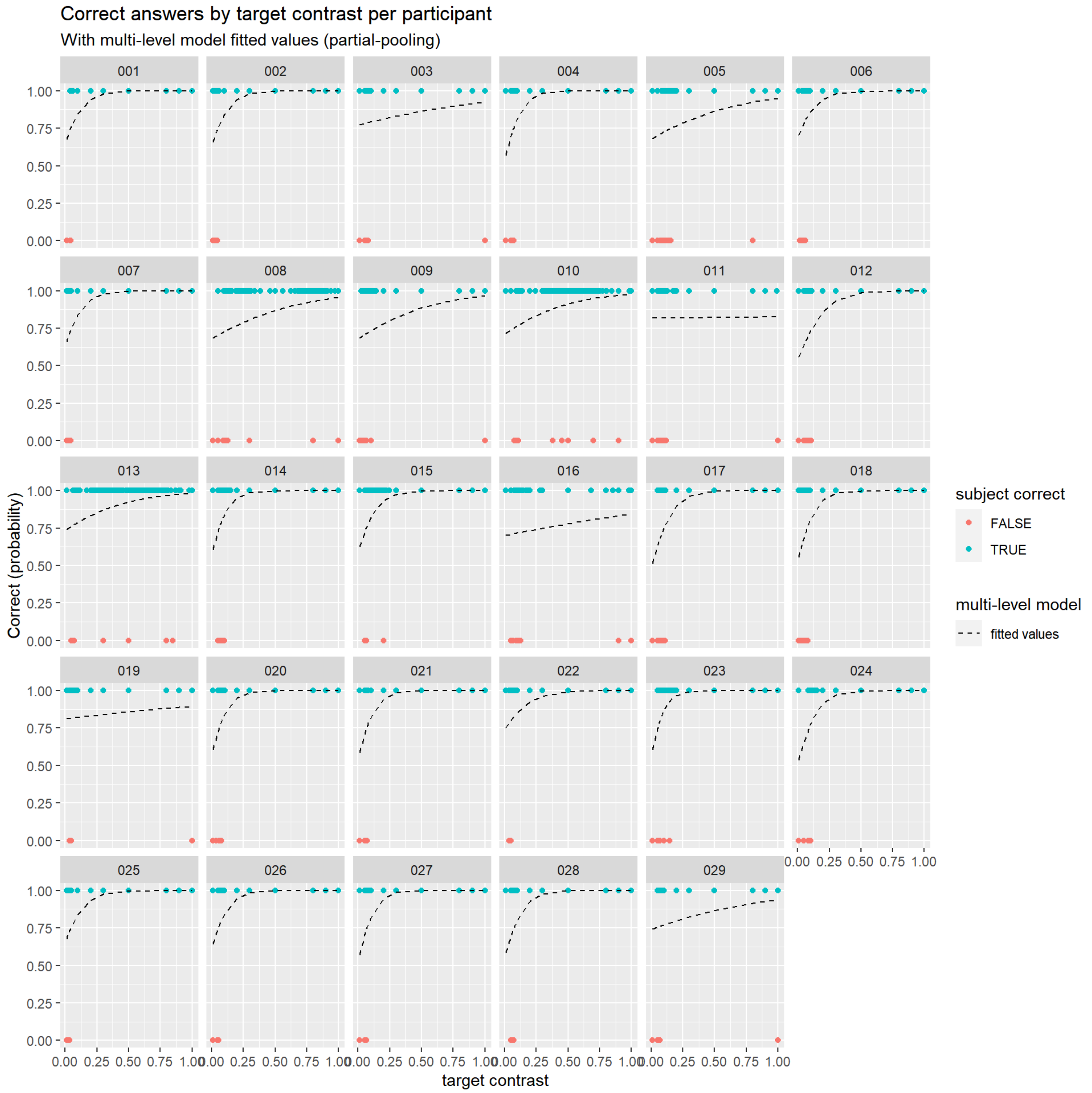
Ex1.p2.iii Comment on the fits - do we have enough data to plot the logistic functions?

The fits are not great. We can see that the lower contrast seems to result in more incorrect answers but even so, subjects still get correct answer at low contrast. As this is a logical outcome variable, the probabilities for most subjects start around 75%. If the contrast is 0.00, we would expect that the subject would not be able to correctly identify the target stimulus, and should start at 50% (chance level). It does seem as though there is sufficient data, the minimum number of samples for a subject is 70, although there are not the same number of samples for each contrast value.

```
m2 <- glmer(correct ~ target.contrast + (1 + target.contrast | subject),
    data = samples_staircase,
    family = binomial(link = "logit"))
summary(m2)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ target.contrast + (1 + target.contrast | subject)
##    Data: samples_staircase
##
##      AIC      BIC   logLik deviance df.resid
##   5988.5   6021.6  -2989.2   5978.5     5598
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.7671  0.0068  0.5532  0.5915  0.9264
##
## Random effects:
##  Groups  Name            Variance Std.Dev. Corr
##  subject (Intercept)      0.2717  0.5213
##          target.contrast 42.7577  6.5389   -0.84
## Number of obs: 5603, groups:  subject, 29
##
## Fixed effects:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)       0.5619     0.1580   3.557 0.000376 ***
## target.contrast   8.7132     2.3604   3.691 0.000223 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## trgt.cntrst -0.907
```

```
samples_staircase %>%
    ggplot(aes(target.contrast, as.integer(correct), color = correct)) +
    geom_point() +
    geom_line(aes(target.contrast, fitted(m2),
        linetype = "fitted values"), inherit.aes = FALSE) +
    scale_linetype_manual(name = "multi-level model", values = c("dashed")) +
    facet_wrap(~subject) +
    labs(title = "Correct answers by target contrast per participant",
        subtitle = "With multi-level model fitted values (partial-pooling)",
        color = "subject correct") +
    ylab("Correct (probability)") +
    xlab("target contrast")
```



Ex1.p2. v. in your own words, describe how the partial pooling model allows for a better fit for each subject

It allows for per-subject differences in their response to the target contrast. If some subjects have more difficulty seeing differences in low contrast images, we would expect their accuracy to decrease more than other subjects as the contrast values becomes lower. We would conceptually expect no-pooling models to create a better fit for each subject(?), as the subjective models are modelled solely on a given subject. However we would of course always choose the partial pooling models as these would give us great subject-fits but also a model generalizabel for the population.

Exercise 2

Now we **only** look at the *experiment* trials (*trial.type*)

1. Pick four subjects and plot their Quantile-Quantile (Q-Q) plots for the residuals of their objective response times (*rt.obj*) based on a model where only intercept is modelled
    i. comment on these
    ii. does a log-transformation of the response time data improve the Q-Q-plots?
2. Now do a partial pooling model modelling objective response times as dependent on *task*? (set `REML=FALSE` in your `lmer`-specification)
    i. which would you include among your random effects and why? (support your choices with relevant measures, taking into account variance explained and number of parameters going into the modelling)
    ii. explain in your own words what your chosen models says about response times between the different tasks
3. Now add *pas* and its interaction with *task* to the fixed effects
    i. how many types of group intercepts (random effects) can you add without ending up with convergence issues or singular fits?
    ii. create a model by adding random intercepts (without modelling slopes) that results in a singular fit - then use `print(VarCorr(<your.model>), comp='Variance')` to inspect the variance vector - explain why the fit is singular (Hint: read the first paragraph under details in the help for `isSingular`)
    iii. in your own words - how could you explain why your model would result in a singular fit?

Answers

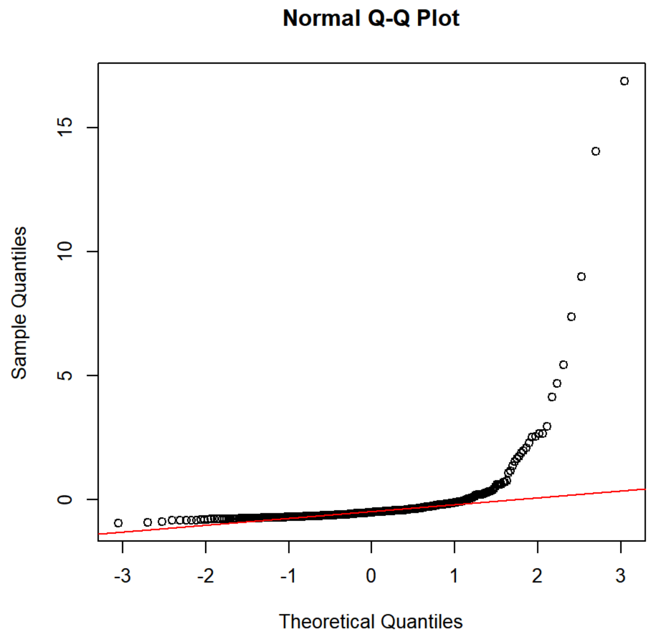
Exercise 2, part 1 - KV

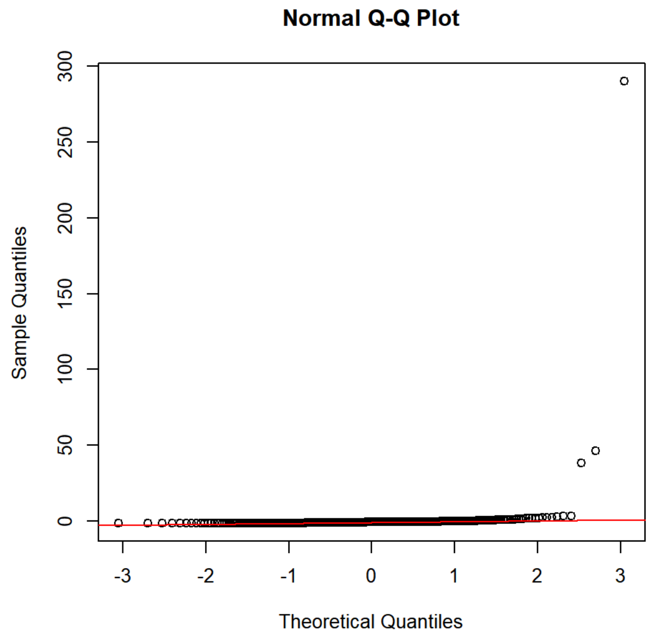
We will use subjects 002, 009, 014 and 015.

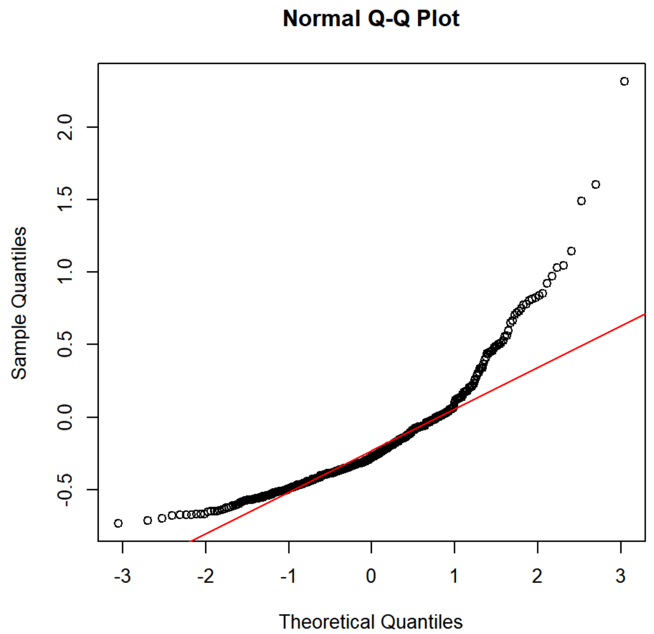
```r
samples_experiment <- samples %>% filter(
    trial.type == "experiment" &
    (
        subject == "002" |
        subject == "009" |
        subject == "014" |
        subject == "015"
    ))

m3 <- lmer(rt.obj ~ (1 | subject),
    data = samples_experiment)
summary(m3)
```

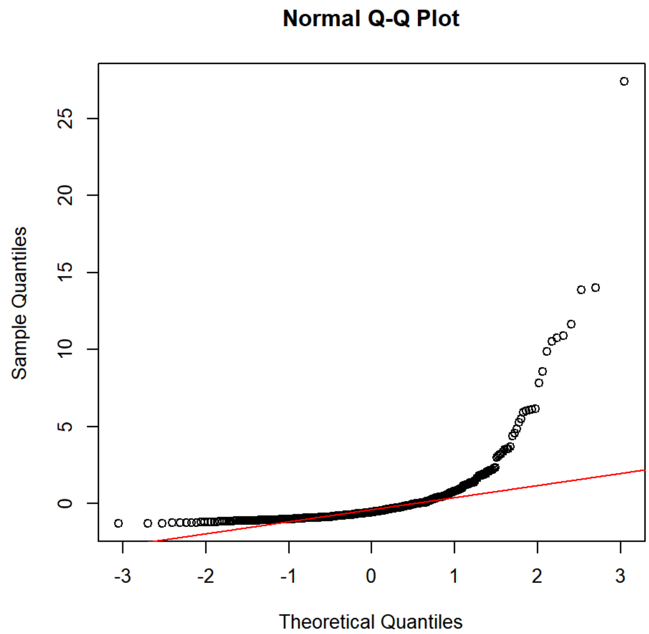
```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: rt.obj ~ (1 | subject)
##    Data: samples_experiment
##
## REML criterion at convergence: 11774.5
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -0.214 -0.098 -0.063 -0.019 39.793
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  subject  (Intercept)  0.1621  0.4026
##  Residual             53.2091  7.2945
## Number of obs: 1728, groups:  subject, 4
##
## Fixed effects:
##             Estimate Std. Error    df t value Pr(>|t|)
## (Intercept)    1.239      0.267 3.000    4.64   0.0189 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
samples_experiment$m3_resid <- residuals(m3)

# make 2x2 plot grid
par(mfrow = c(2, 2))
# plot qq plots for objective response time of subjects:
# 002
qqnorm(samples_experiment[samples_experiment$subject == "002", ]$m3_resid)
qqline(samples_experiment[samples_experiment$subject == "002", ]$m3_resid,
    col = "red")
# 009
qqnorm(samples_experiment[samples_experiment$subject == "009", ]$m3_resid)
qqline(samples_experiment[samples_experiment$subject == "009", ]$m3_resid,
    col = "red")
# 014
qqnorm(samples_experiment[samples_experiment$subject == "014", ]$m3_resid)
qqline(samples_experiment[samples_experiment$subject == "014", ]$m3_resid,
    col = "red")
# 015
qqnorm(samples_experiment[samples_experiment$subject == "015", ]$m3_resid)
qqline(samples_experiment[samples_experiment$subject == "015", ]$m3_resid,
    col = "red")
```









Ex2.p1.i

These data show a significant tail with some reaction times going far beyond the mean (e.g. 15 seconds +).

```
m3l <- lmer(log(rt.obj) ~ (1 | subject),
    data = samples_experiment)
summary(m3l)
```

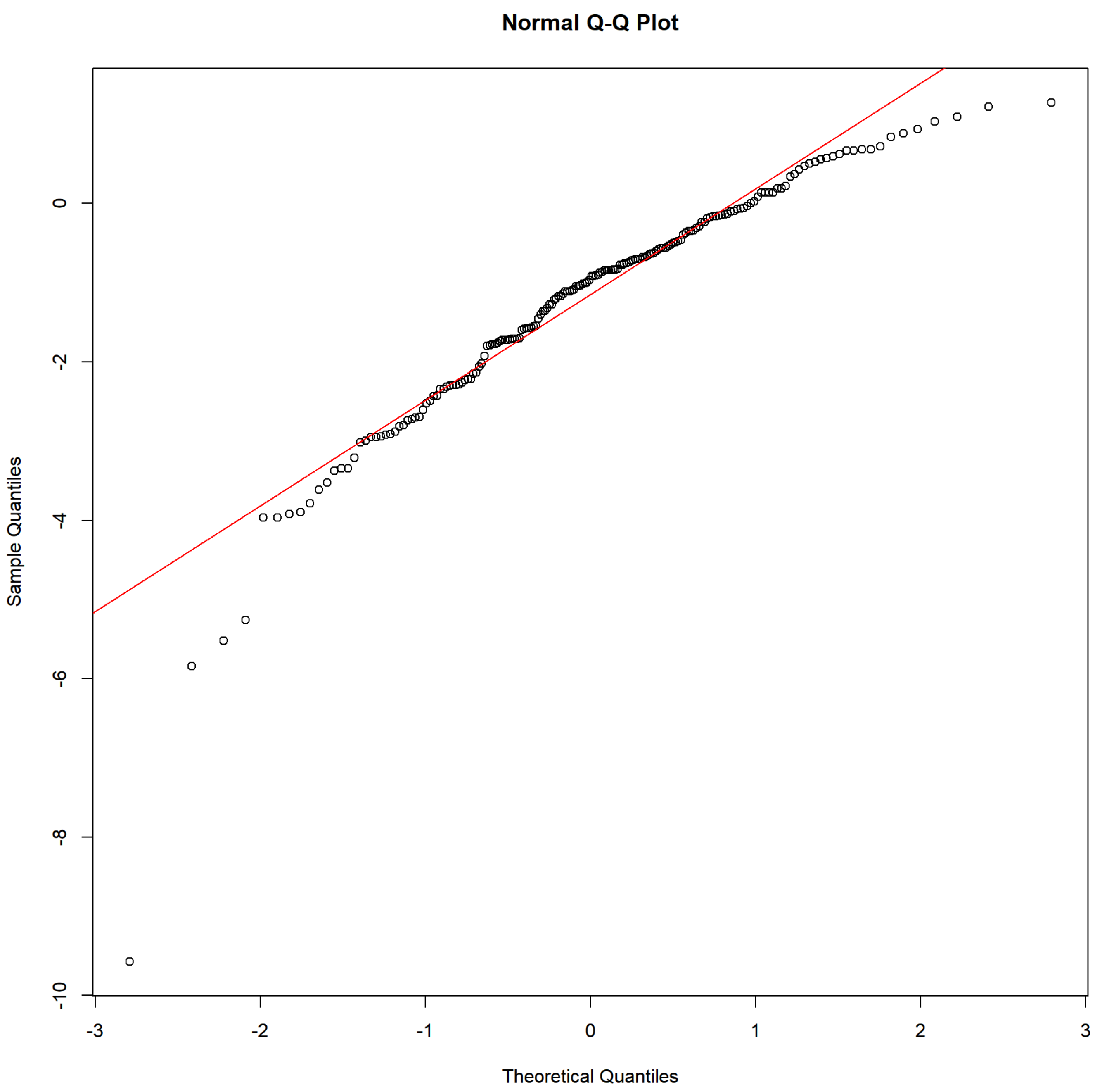
```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: log(rt.obj) ~ (1 | subject)
##    Data: samples_experiment
##
## REML criterion at convergence: 4775.9
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -8.4491 -0.4344  0.0168  0.4933  6.4941
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  subject  (Intercept) 0.08853  0.2975
##  Residual             0.92008  0.9592
## Number of obs: 1728, groups:  subject, 4
##
## Fixed effects:
##             Estimate Std. Error      df t value Pr(>|t|)
## (Intercept)  -0.3916     0.1505  3.0000  -2.601   0.0803 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
samples_experiment$m3l_resid <- residuals(m3l)

# plot qq plots for log transformed objective response time of subjects:
# 002
qqnorm(log(samples_experiment[samples_experiment$subject == "002", ]$m3l_resid))
```

```
## Warning in log(samples_experiment[samples_experiment$subject == "002", ]
## $m3l_resid): NaNs produced
```

```
qqline(log(samples_experiment[samples_experiment$subject == "002", ]$m3l_resid),
    col = "red")
```

```
## Warning in log(samples_experiment[samples_experiment$subject == "002", ]
## $m3l_resid): NaNs produced
```

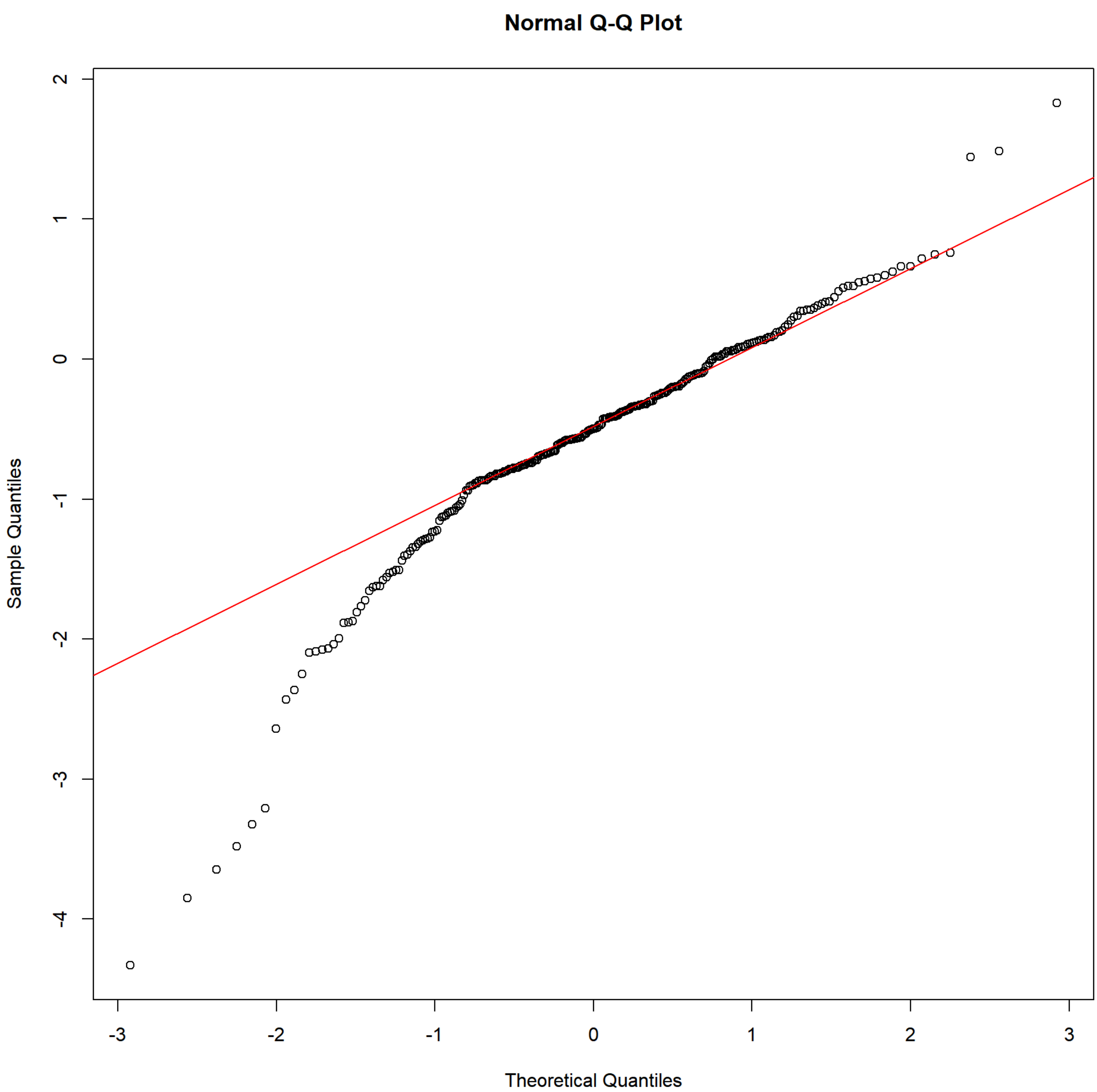



```
# 009
qqnorm(log(samples_experiment[samples_experiment$subject == "009", ]$m3l_resid))
```

```
## Warning in log(samples_experiment[samples_experiment$subject == "009", ]
## $m3l_resid): NaNs produced
```

```
qqline(log(samples_experiment[samples_experiment$subject == "009", ]$m3l_resid),
    col = "red")
```

```
## Warning in log(samples_experiment[samples_experiment$subject == "009", ]
## $m3l_resid): NaNs produced
```

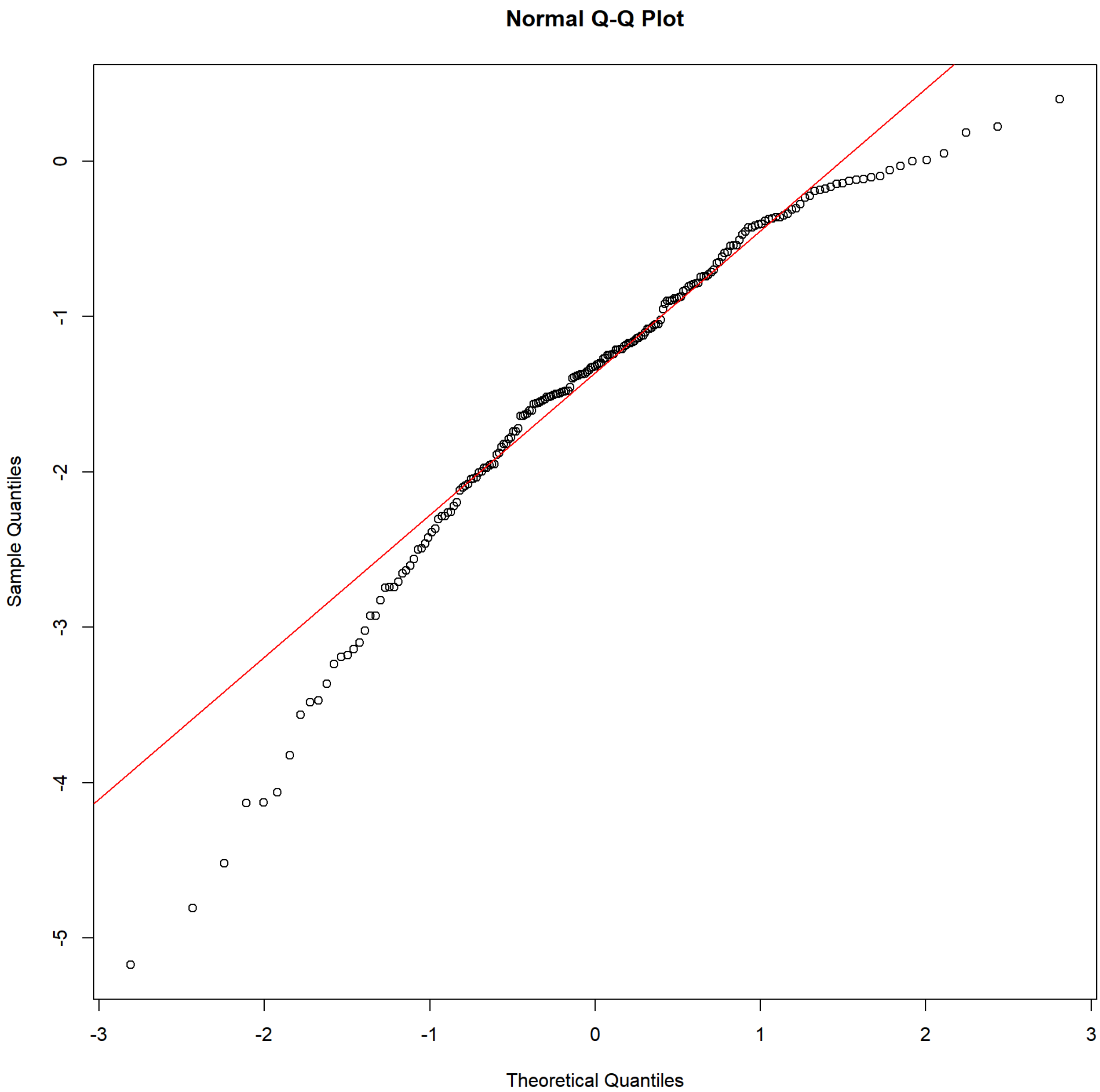



```
# 014
qqnorm(log(samples_experiment[samples_experiment$subject == "014", ]$m3l_resid))
```

```
## Warning in log(samples_experiment[samples_experiment$subject == "014", ]
## $m3l_resid): NaNs produced
```

```
qqline(log(samples_experiment[samples_experiment$subject == "014", ]$m3l_resid),
    col = "red")
```

```
## Warning in log(samples_experiment[samples_experiment$subject == "014", ]
## $m3l_resid): NaNs produced
```

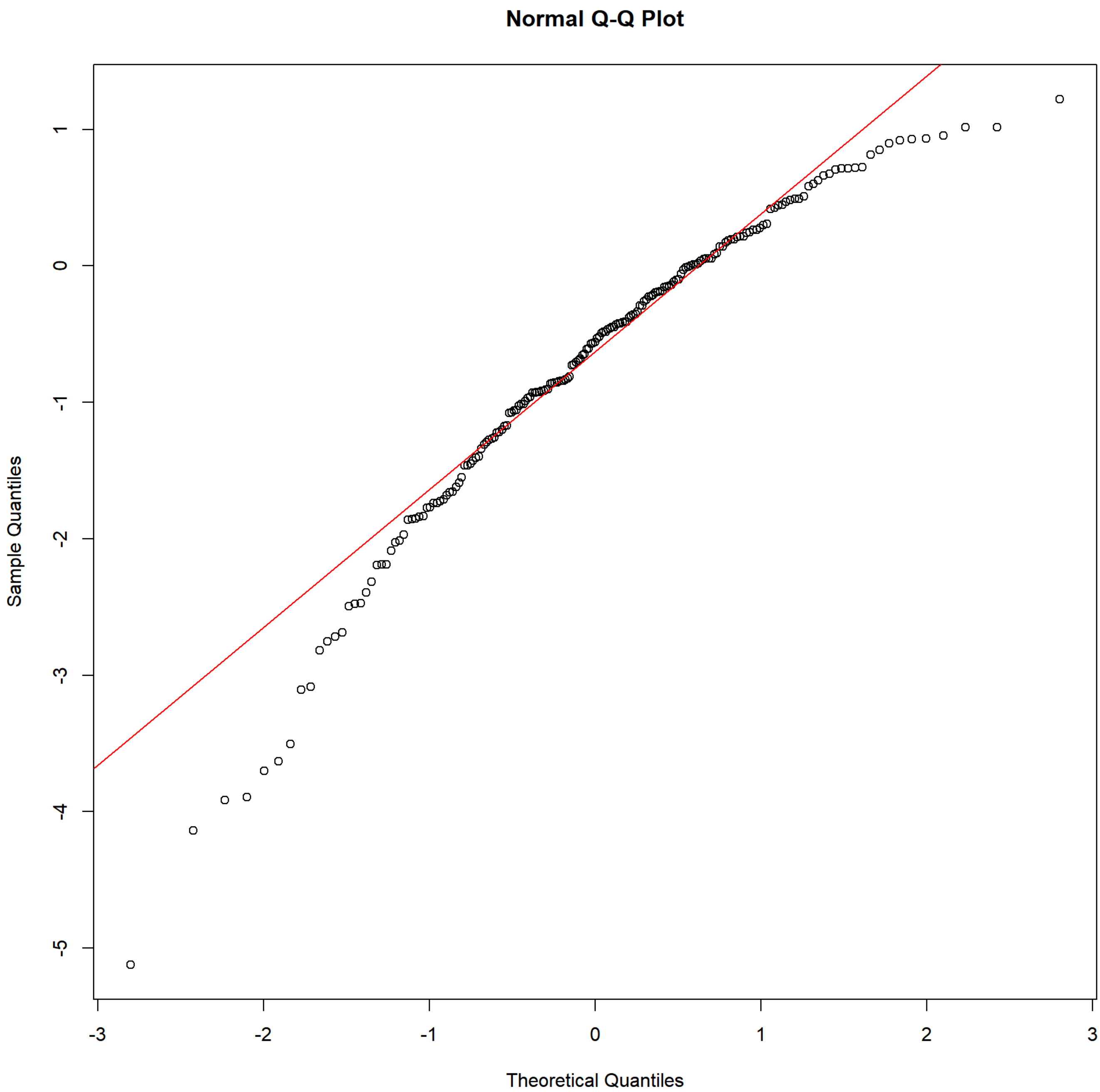



```
# 015
qqnorm(log(samples_experiment[samples_experiment$subject == "015", ]$m3l_resid))
```

```
## Warning in log(samples_experiment[samples_experiment$subject == "015", ]
## $m3l_resid): NaNs produced
```

```
qqline(log(samples_experiment[samples_experiment$subject == "015", ]$m3l_resid),
    col = "red")
```

```
## Warning in log(samples_experiment[samples_experiment$subject == "015", ]
## $m3l_resid): NaNs produced
```




```
# reset plot grid
par(mfrow = c(1, 1))
```

Ex2.p1.ii

Log-transforming the data definitely creates a better fit, as can be seen in the qq-plots.

Exercise 2, part 2 - VK

```
m4 <- lmer(rt.obj ~ task + (1 | subject),
    REML = FALSE, data = samples_experiment)
summary(m4)
```

```
## Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
##   method [lmerModLmerTest]
## Formula: rt.obj ~ task + (1 | subject)
##    Data: samples_experiment
##
##      AIC      BIC   logLik deviance df.resid
##  11780.8  11808.0  -5885.4  11770.8     1723
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -0.258 -0.111 -0.058 -0.010 39.780
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  subject  (Intercept)  0.09096 0.3016
##  Residual             53.12390 7.2886
## Number of obs: 1728, groups:  subject, 4
##
## Fixed effects:
##                  Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)        1.6487     0.3391   18.4270   4.863 0.000117 ***
## taskquadruplet   -0.6554     0.4295 1724.0000  -1.526 0.127187
## tasksingles      -0.5736     0.4295 1724.0000  -1.336 0.181883
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr
## taskqudrplt -0.633
## tasksingles -0.633  0.500
```

```
piecewiseSEM::rsquared(m4)
```

```
##   Response   family     link method    Marginal Conditional
## 1   rt.obj gaussian identity   none 0.001596197  0.00330281
```

2.2.i. which would you include among your random effects and why?

The above model uses random intercepts for subject. We considered that task and trial might be good additional random effects, but these result in a singular fit. We considered task to be a relevant random effect, as it was expected that the different tasks would influence the reaction time. We considered trial to be a relevant random effect as participants might perform better throughout trials.

Overall this model does not explain much of the variance (0.33% about 0.16% fixed effects and 0.17% random effects). The model shows no statistically significant difference between response times per task.

Exercise 2, part 3 - LR

```
# model with PAS-task interaction.
m5 <- lmer(rt.obj ~ task + pas:task + (1 | subject),
    REML = FALSE, data = samples_experiment)
summary(m5)
```

```
## Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
##   method [lmerModLmerTest]
## Formula: rt.obj ~ task + pas:task + (1 | subject)
##    Data: samples_experiment
##
##      AIC      BIC   logLik deviance df.resid
##  11785.3  11828.9  -5884.6  11769.3     1720
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -0.298 -0.107 -0.053 -0.006 39.756
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  subject  (Intercept)  0.08422 0.2902
##  Residual             53.08183 7.2857
## Number of obs: 1728, groups:  subject, 4
##
## Fixed effects:
##                      Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)           2.38869    0.73777  284.85013   3.238  0.00135 **
## taskquadruplet       -1.54762    1.03036 1727.71994  -1.502  0.13328
## tasksingles          -1.03349    1.01117 1727.59916  -1.022  0.30689
## taskpairs:pas        -0.42621    0.37819 1539.29789  -1.127  0.25993
## taskquadruplet:pas    0.09117    0.40230 1716.78531   0.227  0.82075
## tasksingles:pas      -0.15525    0.35936 1174.87743  -0.432  0.66581
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr tsksng tskpr: tskqd:
## taskqudrplt -0.685
## tasksingles -0.693  0.497
## taskpars:ps -0.890  0.633  0.639
## tskqdrplt:p -0.005 -0.648 -0.002  0.006
## tsksngls:ps -0.014  0.004 -0.631  0.015  0.010
```

Ex2.p3.i

Any additional random effects we added besides subject resulted in singular fits.

```
# here already it becomes a singular fit
m6 <- lmer(rt.obj ~ task + pas:task + (1 | subject) + (1 | task),
    REML = FALSE, data = samples_experiment)
```

```
## boundary (singular) fit: see ?isSingular
```

```
summary(m6)
```

```
## Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
##   method [lmerModLmerTest]
## Formula: rt.obj ~ task + pas:task + (1 | subject) + (1 | task)
##    Data: samples_experiment
##
##      AIC      BIC   logLik deviance df.resid
##  11787.3  11836.4  -5884.6  11769.3     1719
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -0.298 -0.107 -0.053 -0.006 39.756
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  subject  (Intercept)  0.08422 0.2902
##  task     (Intercept)  0.00000 0.0000
##  Residual             53.08183 7.2857
## Number of obs: 1728, groups:  subject, 4; task, 3
##
## Fixed effects:
##                       Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)            2.38869    0.73777  284.85013   3.238  0.00135 **
## taskquadruplet        -1.54762    1.03036 1727.71994  -1.502  0.13328
## tasksingles           -1.03349    1.01117 1727.59916  -1.022  0.30689
## taskpairs:pas         -0.42621    0.37819 1539.29788  -1.127  0.25993
## taskquadruplet:pas     0.09117    0.40230 1716.78531   0.227  0.82075
## tasksingles:pas       -0.15525    0.35936 1174.87743  -0.432  0.66581
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr tsksng tskpr: tskqd:
## taskqudrplt -0.685
## tasksingles -0.693  0.497
## taskpars:ps -0.890  0.633  0.639
## tskqdrplt:p -0.005 -0.648 -0.002  0.006
## tsksngls:ps -0.014  0.004 -0.631  0.015  0.010
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see ?isSingular
```

```
print(VarCorr(m6), comp = "Variance")
```

```
##  Groups   Name        Variance
##  subject  (Intercept)  0.084224
##  task     (Intercept)  0.000000
##  Residual             53.081833
```

Ex2.p3.iii. in your own words - how could you explain why your model would result in a singular fit?

In the example above, model m6, it results in a singular fit. Singular fits occur when the variances of some of the linear combinations of the effects are either zero or close to zero, and it is indicating that the model may be overfitting for the data. In model m6, we get the variance for task as 0, which results in a singular fit - this indicates to us that this model is overfitted to the data.

We expect that the reason that the model results in a singular fit is because we have too many parameters, and is therefore too too complex. As mentioned above, adding any additional random effects besides subject resulted in getting singular fits, so in order for the example above, m6, to not result in a singular fit, we would have to remove (1|Task).

Exercise 3

1. Initialise a new data frame, `data.count`. *count* should indicate the number of times they categorized their experience as *pas* 1-4 for each *task*. I.e. the data frame would have for subject 1: for task:singles, pas1 was used # times, pas2 was used # times, pas3 was used # times and pas4 was used # times. You would then do the same for task:pairs and task:quadruplet
2. Now fit a multilevel model that models a unique “slope” for *pas* for each *subject* with the interaction between *pas* and *task* and their main effects being modelled
    i. which family should be used?
    ii. why is a slope for *pas* not really being modelled?
    iii. if you get a convergence error, try another algorithm (the default is the *Nelder_Mead*) - try (*bobyqa*) for which the `dfoptim` package is needed. In `glmer`, you can add the following for the `control` argument: `glmerControl(optimizer="bobyqa")` (if you are interested, also have a look at the function `allFit`)
    iv. when you have a converging fit - fit a model with only the main effects of *pas* and *task*. Compare this with the model that also includes the interaction
    v. indicate which of the two models, you would choose and why
    vi. based on your chosen model - write a short report on what this says about the distribution of ratings as dependent on *pas* and *task*
    vii. include a plot that shows the estimated amount of ratings for four subjects of your choosing
3. Finally, fit a multilevel model that models *correct* as dependent on *task* with a unique intercept for each *subject*
    i. does *task* explain performance?

ii. add *pas* as a main effect on top of *task* - what are the consequences of that?
iii. now fit a multilevel model that models *correct* as dependent on *pas* with a unique intercept for each *subject*
iv. finally, fit a model that models the interaction between *task* and *pas* and their main effects
v. describe in your words which model is the best in explaining the variance in accuracy

Answers

Exercise 3, part 1 - KV

```
data.count <- samples %>% select(subject, pas, task, correct) %>%
    group_by(subject, task, pas) %>%
    summarize(
        subject = subject[1],
        task = task[1],
        pas = pas[1],
        count = n(),
        accuracy = sum(correct) / n(),
        .groups = "drop")
head(data.count)
```

```
## # A tibble: 6 x 5
##   subject task         pas count accuracy
##   <fct>   <fct>      <int> <int>    <dbl>
## 1 001     pairs          1   109    0.532
## 2 001     pairs          2    45    0.756
## 3 001     pairs          3     4    1
## 4 001     pairs          4    12    1
## 5 001     quadruplet     1   141    0.596
## 6 001     quadruplet     2    23    0.652
```

Exercise 3, part 2 - EH

```
# multi-level with slope and interaction
m7 <- glmer(count ~
    pas + task + pas:task +
    (1 + pas | subject),
    data = data.count,
    family = poisson,
    control = glmerControl(optimizer = "bobyqa"))
summary(m7)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: count ~ pas + task + pas:task + (1 + pas | subject)
##    Data: data.count
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##   4685.1   4719.6  -2333.6   4667.1      331
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.7718 -1.9208 -0.1275  1.6133 11.6477
##
## Random effects:
##  Groups  Name        Variance Std.Dev. Corr
##  subject (Intercept) 1.2017   1.0962
##          pas         0.2203   0.4694   -0.99
## Number of obs: 340, groups:  subject, 29
##
## Fixed effects:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)         4.28937    0.20577  20.846  < 2e-16 ***
## pas                -0.19479    0.08798  -2.214   0.0268 *
## taskquadruplet      0.16669    0.04007   4.160 3.18e-05 ***
## tasksingles        -0.39660    0.04192  -9.461  < 2e-16 ***
## pas:taskquadruplet -0.07195    0.01606  -4.480 7.47e-06 ***
## pas:tasksingles     0.16855    0.01587  10.622  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) pas    tskqdr tsksng ps:tskq
## pas         -0.989
## taskqudrplt -0.100  0.083
## tasksingles -0.096  0.078  0.490
## ps:tskqdrpl  0.088 -0.091 -0.891 -0.430
## ps:tsksngls  0.089 -0.091 -0.456 -0.900  0.501
```

Ex3.p2.i

For this model (m7) we used the poisson family because the poisson distribution is specifically used to describe probabilities of event frequencies.

Ex3.p2.ii

There is not really a slope being modelled for PAS as PAS is not continuous, it is a factor, so we only get esimates for each PAS level. As a factor, PAS is treated categorically (PAS1, PAS2, PAS3, PAS4), there is no such thing as PAS 1.3 etc and as such, it can not really be modelled as a slope.

```r
m7.1 <- glmer(count ~
    pas + task + (1 + pas | subject),
    data = data.count,
    family = poisson,
    control = glmerControl(optimizer = "bobyqa"))
summary(m7.1)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: count ~ pas + task + (1 + pas | subject)
##    Data: data.count
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##   4923.2   4950.0  -2454.6   4909.2      333
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.9891 -2.1490 -0.1911  1.8233 11.0534
##
## Random effects:
##  Groups  Name        Variance Std.Dev. Corr
##  subject (Intercept) 1.2147   1.1021
##          pas         0.2232   0.4725   -0.99
## Number of obs: 340, groups:  subject, 29
##
## Fixed effects:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     4.213654   0.205804  20.474   <2e-16 ***
## pas            -0.162486   0.088064  -1.845    0.065 .
## taskquadruplet  0.006593   0.018184   0.363    0.717
## tasksingles     0.004307   0.018189   0.237    0.813
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) pas    tskqdr
## pas         -0.989
## taskqudrplt -0.044  0.000
## tasksingles -0.044  0.000  0.501
```

```r
AIC(m7, m7.1)
```

```
##      df      AIC
## m7    9 4685.119
## m7.1  7 4923.190
```

Ex3.p2.v Indicate which of the two models, you would choose and why?

Between m7 and m7.1, we believe that the m7 is the best model. Out of the two, m7 is the only one thats predictors have any significance on the outcome, in particular, the interactions in m7 have significant effect on the count variable. When comparing the AIC output, it is also shown that model m7 has a better fit, with an AIC of 3148.441, compared to m7.1 of 3398.549. Taking all this into consideration, we are inclined to use this model m7 out of the two as it is the better model.

Ex3.p2.vi based on your chosen model - write a short report on what this says about the distribution of ratings as dependent on pas and task ?

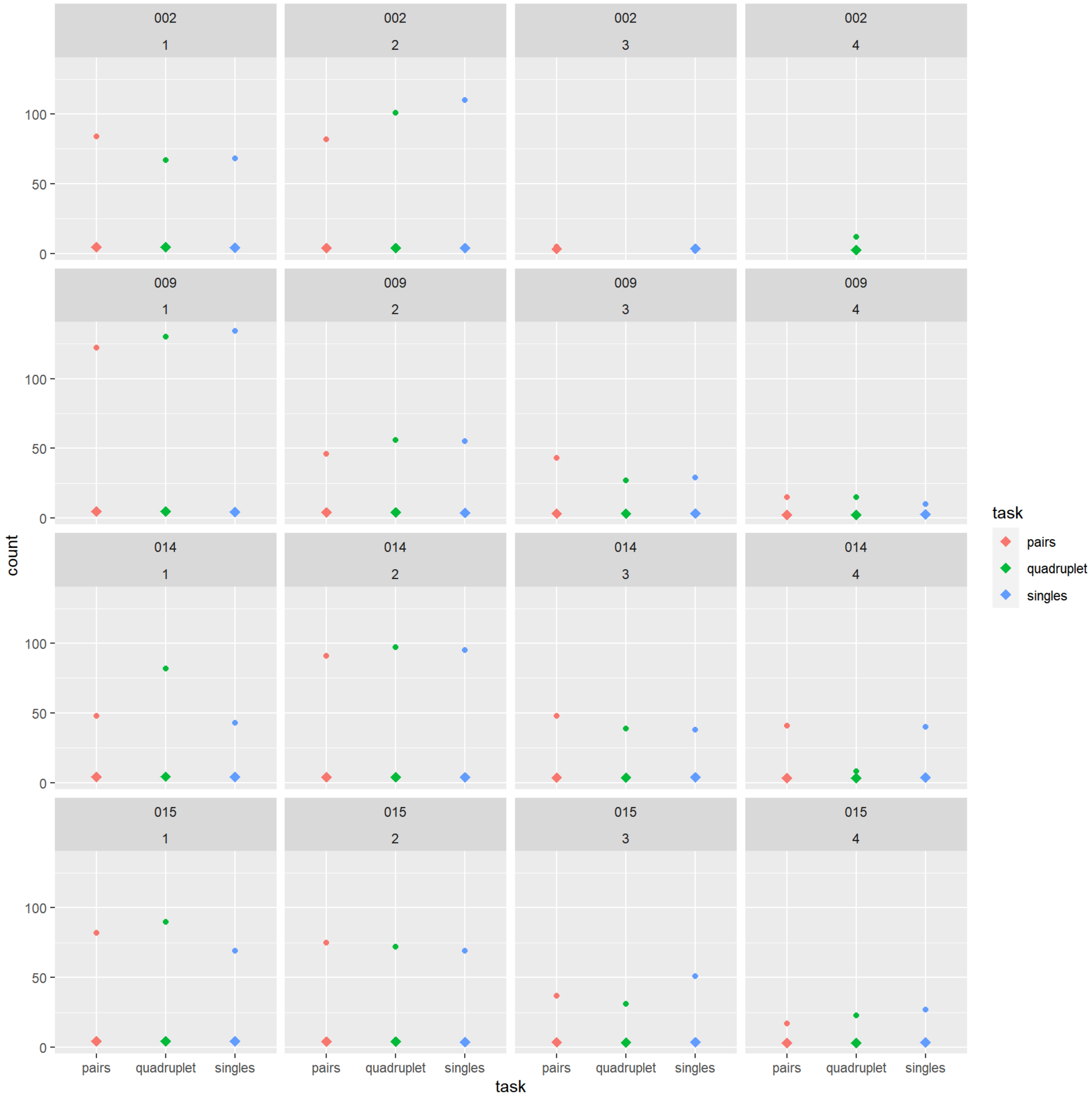
m7 tells us that there is an important interaction between pas and task when it comes to the distribution of ratings, as all interactions reach significance, $p < 0.05$, except for pas3:taskquadruplet, $p = 0.92020$ and pas2:taskquadruplet, $p = 0.06010$

We can see from m7 that there is a decrease of all the interactions between pas and quadruplet tasks (pas:quadruplet), est. -0.0719. We see the opposite in the case all counts of interactions between pass and single tasks, where there is an increase of est. ~0.169.

Looking at pas in m7, there is an increase of count in all pas when going from pas4 to pas1 (0.772932), 2 (est. 0.749155) and 3 (est. 0.259281). Only the predictions for pas1 and pas2 are significant, $p < 0.05$.

Regarding task in our model, count decreases going from task pairs to task quadruplets, est. -0.100, and count increases going from task pairs to task singles, ~ est. 0.333. Both of these predications are significant, $p < 0.05$.

```
# get only specified subjects
data.count_subset <- data.count %>%
    filter(
        subject == "002" |
        subject == "009" |
        subject == "014" |
        subject == "015")

# predict using model for only selected subjects
m7_fitted <- predict(m7, newdata = data.count_subset)

# plot. is this right? the predictions seem way off...
data.count_subset %>% ggplot(aes(task, count, color = task)) +
    geom_point() +
    geom_point(aes(x = task, y = m7_fitted), shape = 18, size = 3) +
    facet_wrap(subject~pas)
```



Exercise 3, part 3 - LR

```
m8 <- glmer(correct ~ task +
    (1 | subject),
    data = samples,
    family = binomial(link = "logit"))
summary(m8)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ task + (1 | subject)
##    Data: samples
##
##      AIC      BIC   logLik deviance df.resid
##  19927.2  19958.4  -9959.6  19919.2    18127
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.7426 -1.0976  0.5098  0.6101  0.9111
##
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.1775   0.4214
## Number of obs: 18131, groups:  subject, 29
##
## Fixed effects:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)     1.10071    0.08386  13.125  < 2e-16 ***
## taskquadruplet -0.09825    0.04190  -2.345    0.019 *
## tasksingles     0.18542    0.04337   4.276 1.91e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr
## taskqudrplt -0.256
## tasksingles -0.247  0.495
```

```
# seems task only explains a tiny amount of performance
piecewiseSEM::rsquared(m8)
```

```
##   Response   family  link method    Marginal Conditional
## 1  correct binomial logit  delta 0.002526141  0.03489787
```

Ex3.p3.i

Differences in the task seem to explain performance (number of correct answers) to a statistically significant level.

```
m9 <- glmer(correct ~ task + pas +
    (1 | subject),
    data = samples,
    family = binomial(link = "logit"))
summary(m9)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ task + pas + (1 | subject)
##    Data: samples
##
##      AIC      BIC   logLik deviance df.resid
##  17425.0  17464.0  -8707.5  17415.0    18126
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -8.1096 -0.6101  0.3181  0.5653  1.6476
##
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.2004   0.4477
## Number of obs: 18131, groups:  subject, 29
##
## Fixed effects:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -0.950104   0.098399  -9.656   <2e-16 ***
## taskquadruplet -0.029418   0.045016  -0.653    0.513
## tasksingles    -0.008914   0.046889  -0.190    0.849
## pas             1.014031   0.022900  44.281   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tskqdr tsksng
## taskqudrplt -0.247
## tasksingles -0.189  0.489
## pas         -0.421  0.030 -0.083
```

```
# better
piecewiseSEM::rsquared(m9)
```

```
##   Response   family  link method  Marginal Conditional
## 1  correct binomial logit  delta 0.1925254   0.2219821
```

Ex3.p3.ii

When we add PAS as a main effect on top of task, we can see that PAS actually has a significant effect on performance, whereas now task no longer displays a significant effect. This means that by including PAS as a main effect in the model, it altered the significance of task on performance.

```
m10 <- glmer(correct ~ pas +
    (1 | subject),
    data = samples,
    family = binomial(link = "logit"))
summary(m10)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ pas + (1 | subject)
##    Data: samples
##
##      AIC      BIC   logLik deviance df.resid
##  17421.5  17444.9  -8707.7  17415.5    18128
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -8.1864 -0.6117  0.3187  0.5664  1.6348
##
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.2005   0.4478
## Number of obs: 18131, groups:  subject, 29
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.96488    0.09504  -10.15   <2e-16 ***
## pas          1.01488    0.02275   44.62   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##     (Intr)
## pas -0.440
```

```
# task makes negligible diff
piecewiseSEM::rsquared(m10)
```

```
##   Response   family  link method  Marginal Conditional
## 1  correct binomial logit  delta 0.1925749   0.2220484
```

```
m11 <- glm(correct ~ pas + task +
    pas:task,
    data = samples,
    family = binomial(link = "logit"))
summary(m11)
```

```
##
## Call:
## glm(formula = correct ~ pas + task + pas:task, family = binomial(link = "logit"),
##     data = samples)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4709  -1.2372   0.4923   0.7704   1.1188
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.70591    0.07006 -10.077   <2e-16 ***
## pas                 0.88428    0.03526  25.081   <2e-16 ***
## taskquadruplet      0.05152    0.09696   0.531    0.595
## tasksingles        -0.10956    0.10248  -1.069    0.285
## pas:taskquadruplet -0.04835    0.04925  -0.982    0.326
## pas:tasksingles     0.07069    0.05017   1.409    0.159
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 20418  on 18130  degrees of freedom
## Residual deviance: 17864  on 18125  degrees of freedom
## AIC: 17876
##
## Number of Fisher Scoring iterations: 5
```

```
piecewiseSEM::rsquared(m11)
```

```
##   Response   family  link     method R.squared
## 1  correct binomial logit nagelkerke       NaN
```

```
m12 <- glmer(correct ~ pas + task +
    pas:task + (1 | subject),
    data = samples,
    family = binomial(link = "logit"))
summary(m12)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: correct ~ pas + task + pas:task + (1 | subject)
##    Data: samples
##
##      AIC      BIC   logLik deviance df.resid
##  17422.8  17477.5  -8704.4  17408.8    18124
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -7.6024 -0.6183  0.3163  0.5755  1.6810
##
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  subject (Intercept) 0.2007   0.448
## Number of obs: 18131, groups:  subject, 29
##
## Fixed effects:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.93319    0.11163  -8.360   <2e-16 ***
## pas                 1.00447    0.03749  26.794   <2e-16 ***
## taskquadruplet      0.05256    0.09943   0.529    0.597
## tasksingles        -0.15466    0.10509  -1.472    0.141
## pas:taskquadruplet -0.04782    0.05053  -0.946    0.344
## pas:tasksingles     0.07731    0.05136   1.505    0.132
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) pas    tskqdr tsksng ps:tskq
## pas         -0.600
## taskqudrplt -0.467  0.628
## tasksingles -0.435  0.583  0.493
## ps:tskqdrpl  0.413 -0.694 -0.893 -0.437
## ps:tsksngls  0.405 -0.680 -0.455 -0.894  0.505
```

```
piecewiseSEM::rsquared(m12)
```

```
##   Response   family  link method  Marginal Conditional
## 1  correct binomial logit  delta 0.1944348   0.2238636
```

```
AIC(m8, m9, m10, m12)
```

```
##     df      AIC
## m8   4 19927.21
## m9   5 17425.01
## m10  3 17421.46
## m12  7 17422.82
```

Ex3.p3.v Describe in your words which model is the best in explaining the variance in accuracy?

Of the above models, m12 is the best at explaining the variance in accuracy. But we think that model 10 is a bettter model to use because the difference in explained variance is negligible and it is a simpler model which means it is less prone to overfitting. Adding the random intercepts per subject allows for differences in subjects ability to correctly classify the stimuli. These difference in subjects may come from situational stress. Model m10 also has the lowest AIC score (17421.39), even though it has slightly more residual deviance than model 12, but this makes sense as more complex models are punished in this area.

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)

Student numbers

Emma Rose Hahn: 202004249 Viara Krasteva: 202005673 Kristian Nøhr Villebro: 202005289 Luke Ring: 202009983

Portfolio Assignment 2 part 2

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/00b4953b4108d73427000(Methods-for-Multilevel-Modelling.pdf (https://www.researchgate.net/profile/Douglas-Bates/publication/2753537_Computational_Methods_for_Multilevel_Modelling/links/00b4953b4108d73427000(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

```r
#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
```

```r
#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
```

```r
#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)
```

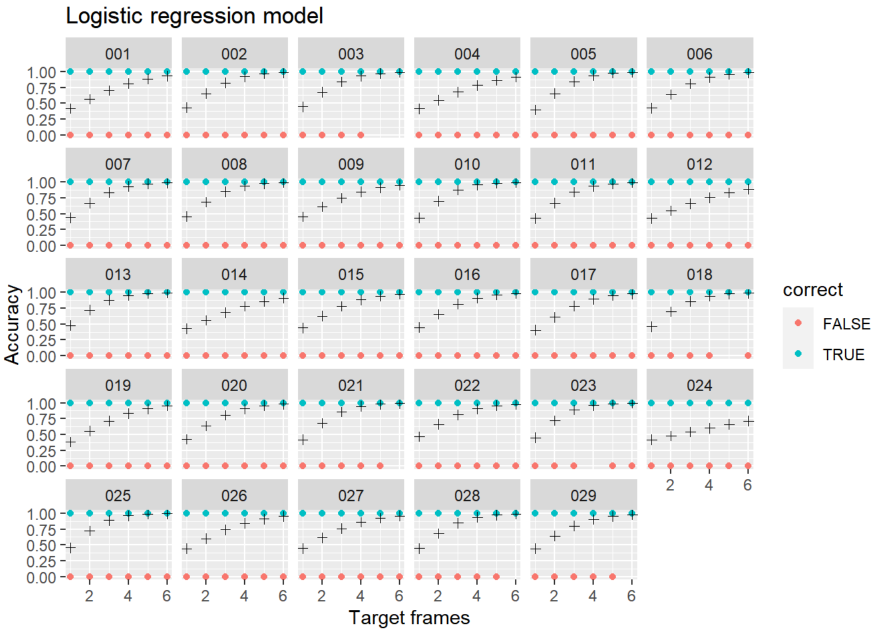
```
(m3_ll <- logLik(m3))
```

```
## '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)
```

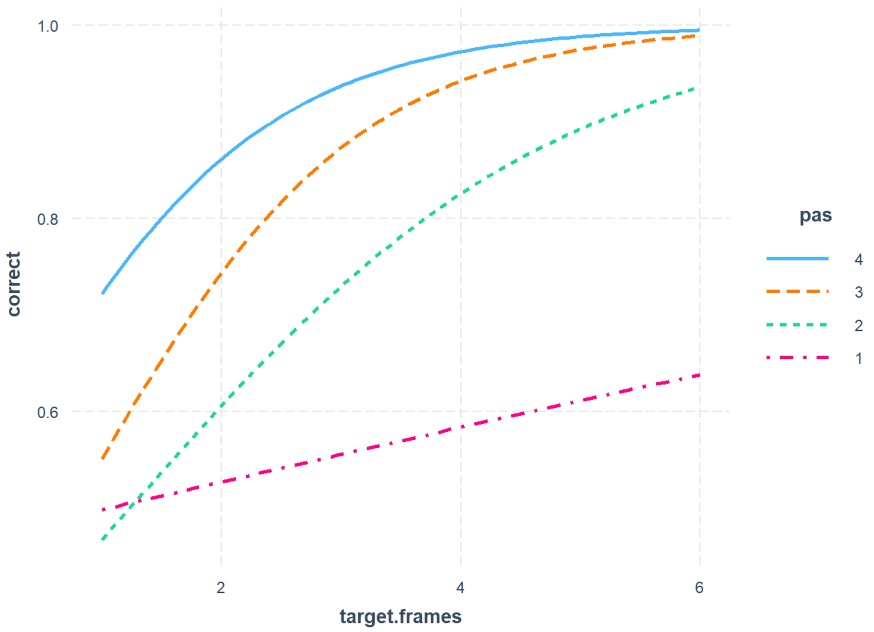
```
m6 <- glmer(correct ~ target.frames + pas +
    (pas * target.frames) + (target.frames | subject),
    data = exp1, family = binomial(link = "logit"), control = glmerControl(optimizer = "bobyqa"))
(m6_ll <- logLik(m6))
```

```
## '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
)
```

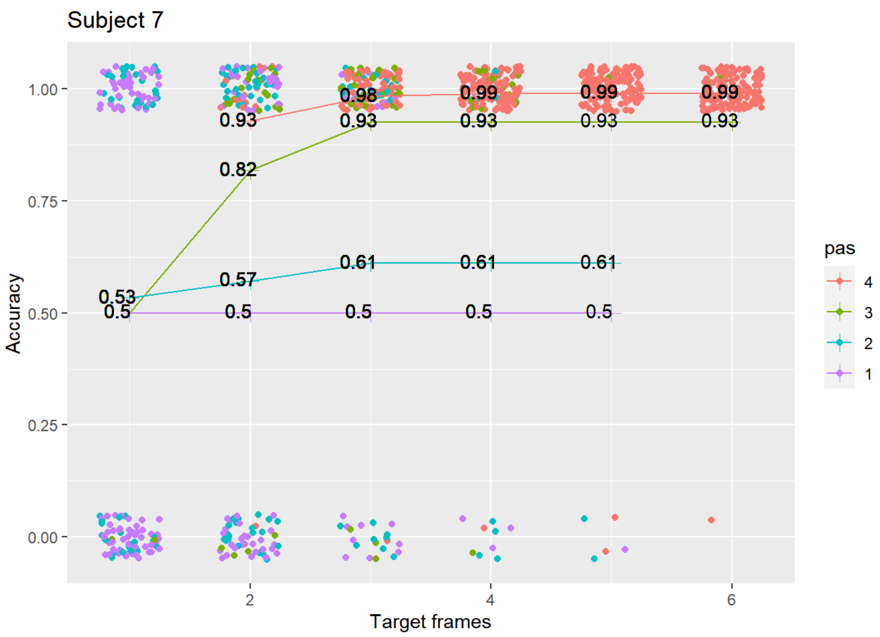
```
length(subj7xy[subj7xy$pas == "1", ]$x)
```

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

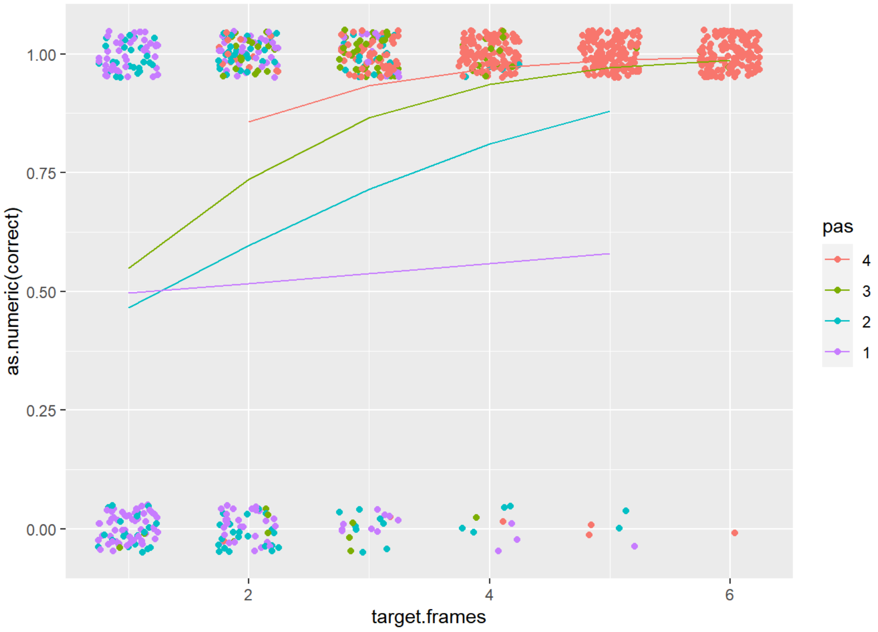
```
length(p1fit)
```

```
## [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))
```

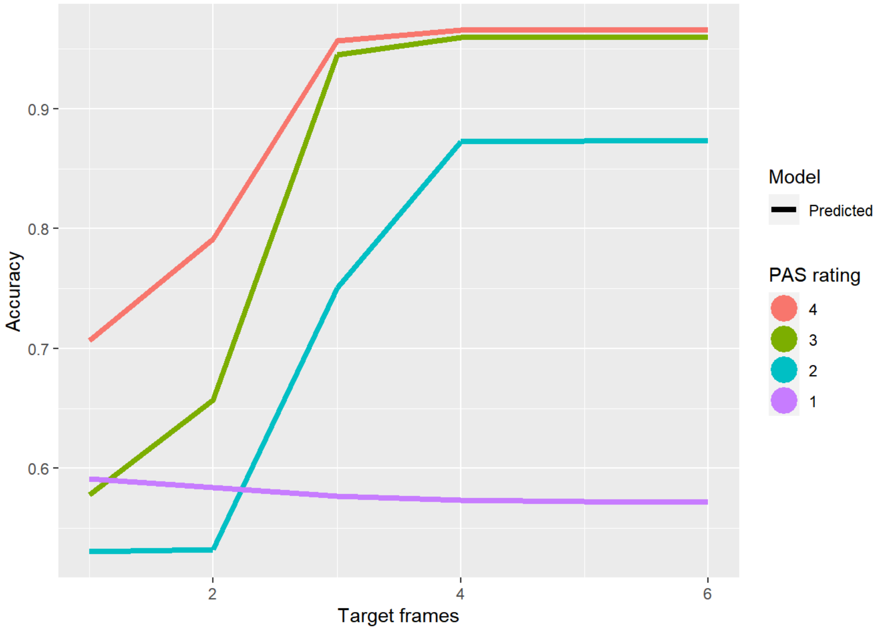


7.1.iv As we plotted the predicted values of the fits, the intercepts at *target.frames*=0 is not shown, though this would have been a crucial difference between the plots.The visible difference between the plots must primarily stem from the fact that we manually set the lowest possible value to 0.5 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.

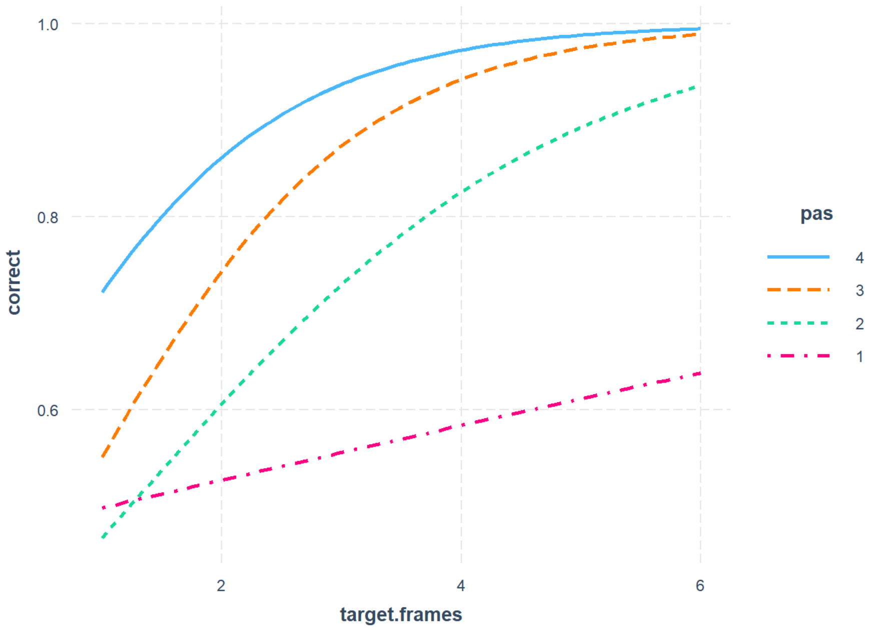
Exercise 7, part 2 - EH

```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"
    )
```

```
## Warning: Computation failed in `stat_summary()`:
## The `Hmisc` package is required.
```




```
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.

Portfolio Assignment 3

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

Student numbers

- Emma Rose Hahn: 202004249
- Kristian Nøhr Villebro: 202005289
- Luke Ring: 202009983
- Viara Krasteva: 202005673

In [ ]:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, ConfusionMatrixDisplay
import seaborn as sns
```

Exercise 1

1.1 Describe data structure (EH)

In [ ]:

```
data = np.load('megmag_data.npy')
dshape  = data.shape
print('Megmag data shape: {}'.format(dshape))
```

```
Megmag data shape: (682, 102, 251)
```

1.1.i There are 682 repetitions, 102 sensors and 251 timestamps

1.1.ii) Add time offset (ms) (EH)

In [ ]:

```
times = np.arange(-200, 801, 4)
print('Time offset array shape: {}'.format(times.shape))
```

```
Time offset array shape: (251,)
```

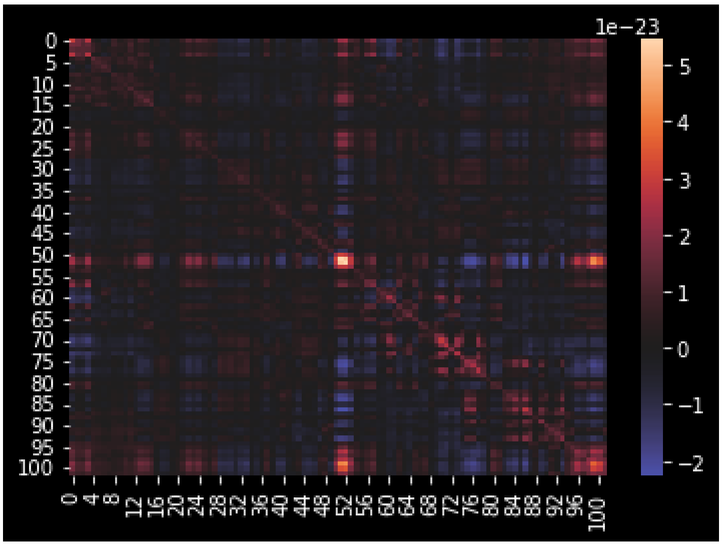
1.1.iii) Create covariance matrix, do the sensors pick up independent signals? (VK)

In [ ]:

```
output = []
for i in range(len(data)):
  X = data[i]
  Xt = X.T
  output.append(np.matrix(np.dot(X,Xt)))
cov_mat = 1  / dshape[0] * np.sum(i for i in output)
```

```
print('Covariance matrix shape: {}'.format(cov_mat.shape))
sns.heatmap(cov_mat, cmap='icefire', center = 0)
# plt.imshow(cov_mat)
plt.show()
```

```
C:\Users\webma\AppData\Local\Temp/ipykernel_9440/1960884245.py:6: Deprecation
Warning: Calling np.sum(generator) is deprecated, and in the future will give
a different result. Use np.sum(np.fromiter(generator)) or the python sum buil
tin instead.
  cov_mat = 1  / dshape[0] * np.sum(i for i in output)
Covariance matrix shape: (102, 102)
```



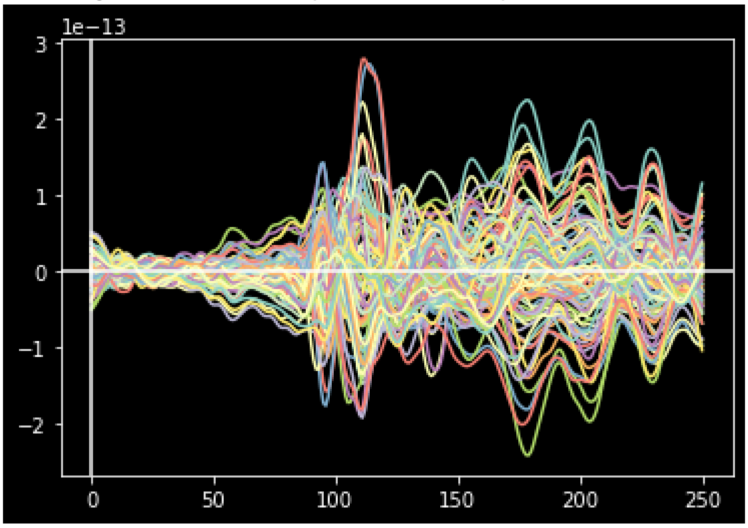
1.1.iii It looks like there is a lot of covariance (which would make sense), however it is quite difficult to get a clear impression with such big covariance matrix.

1.1.iv - 1.1.v) Create signal averages across repititions (KV)

In [ ]:

```
avg_for_reps = np.mean(data, axis=0).T
print('Average for each repetition shape: {}'.format(avg_for_reps.shape))
plt.plot(avg_for_reps)
plt.axvline()
plt.axhline()
plt.show()
```

```
Average for each repetition shape: (251, 102)
```



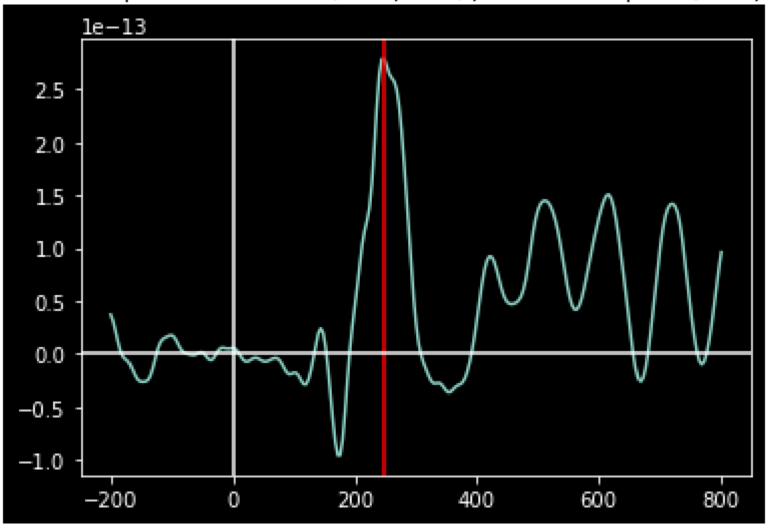
1.1.vi - 1.1.vii Find and plot maximum response channel (KV)

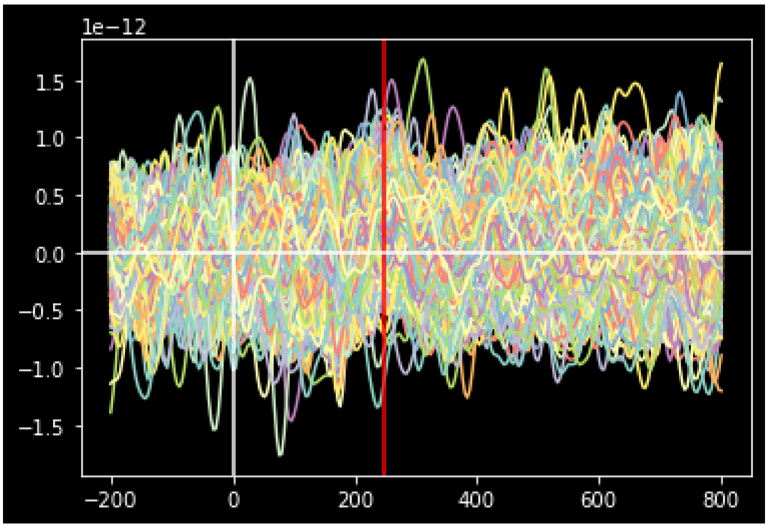
In [ ]:

```
max_resp = np.unravel_index(np.argmax(avg_for_reps), avg_for_reps.shape)
```

```
print('Max response index: {}, data shape: {}'.format(max_resp,
data.shape))
#plot average for all repititions
plt.plot(times, avg_for_reps[:, max_resp[1]])
plt.axvline()
plt.axvline(x = times[max_resp[0]], color='r', label='max response')
plt.axhline()
plt.show()

# plot all repititions for max response channel
plt.plot(times, data[:, max_resp[1], :].squeeze().T)
plt.axvline()
plt.axvline(x = times[max_resp[0]], color='r', label='max response')
plt.axhline()

plt.show()
```

```
Max response index: (112, 73), data shape: (682, 102, 251)
```





1.1.vi We found that the maximum response signal was received by sensor 74 (as python index starts at 0)

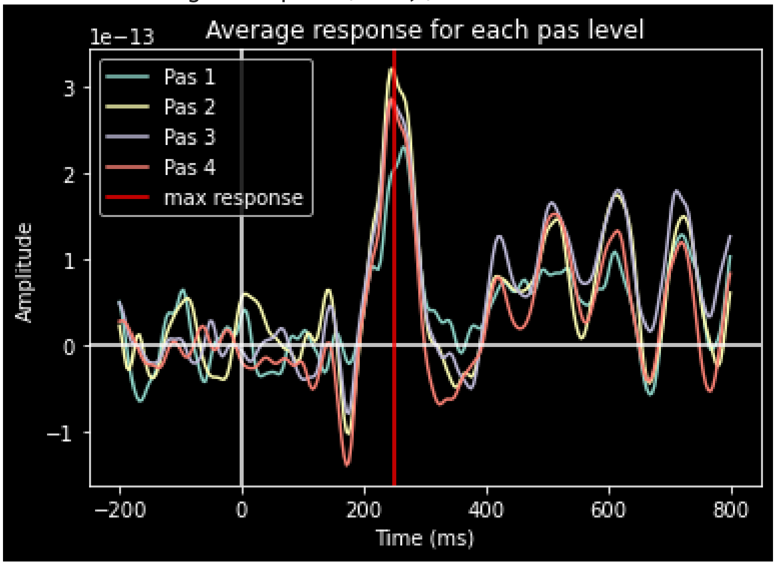
1.1.viii The messy plot showing the single repetitions includes a lot of fluctuation around the y-axis, thus including lots of noise, making it difficult to assess anything from. Even though the messy plot looks as if the activity is more or less the same across the whole time period, the averaged plot reveals that activation is quite different over time. As the signal noise goes in both negative and positive direction, they cancel each other out thus resulting in the averaged plot we have seen.

1.2) load pas_vector.npy (LR)

In [ ]:

```
y = np.load('pas_vector.npy')
# same as number of repititions
print('Pas vector shape: {}'.format(y.shape))
pas_data = []
pas_index = {}
for i in np.unique(y):
  print('Creating pas slice for pas {}'.format(i))
  pas_slice = np.argwhere(y == i)
  data_slice = data[pas_slice].squeeze()
  pas_data.append({
    'paslevel': i,
    'data': data_slice,
  })
  pas_index[i] = len(pas_data) - 1
for i in pas_data:
  print('Pas {} shape: {}'.format(i['paslevel'], i['data'].shape))
  avg_for_pas = np.mean(i['data'][:,max_resp[1],:], axis=0).T
  print('Pas {} average shape: {}'.format(i['paslevel'],
avg_for_pas.shape))
  plt.plot(times, avg_for_pas, label='Pas {}'.format(i['paslevel']))
plt.axvline()
plt.axhline()
plt.axvline(x = times[max_resp[0]], color='r', label='max response')
plt.legend(loc="upper left")
plt.title('Average response for each pas level')
plt.xlabel('Time (ms)')
plt.ylabel('Amplitude')
plt.show()
```

```
Pas vector shape: (682,)
Creating pas slice for pas 1
Creating pas slice for pas 2
Creating pas slice for pas 3
Creating pas slice for pas 4
Pas 1 shape: (99, 102, 251)
Pas 1 average shape: (251,)
Pas 2 shape: (115, 102, 251)
Pas 2 average shape: (251,)
Pas 3 shape: (208, 102, 251)
Pas 3 average shape: (251,)
Pas 4 shape: (260, 102, 251)
Pas 4 average shape: (251,)
```



1.2.i Which dimension in the data array does it have the same length as?

The pas_vector has length 682, thus having the same length as repetitions.

1.2.iii Describe how the amplitudes of responses are related to the four PAS-scores. Does PAS 2 behave differently than expected?

One would expect that the lower the PAS level the lower the response signal – e.g., PAS 1, not perceiving anything, would result in a low signal, and PAS 4 would result in the highest signal. This seems to correlate with what we see in plot. However, surprisingly PAS 2 has a higher max at 250ms than PAS 4 (which we would expect would have the highest).

Exercise 2 - Do logistic regression to classify pairs of PAS-ratings

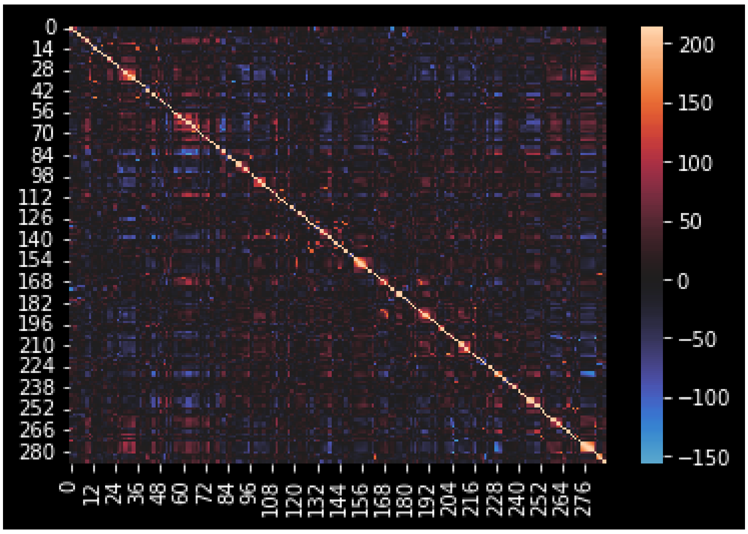
2.1 Logistic regression (EH)

In [ ]:

```
# create a new array called data_1_2 that only contains PAS responses 1 and
2
data_1_2 = np.concatenate((pas_data[pas_index[1]]['data'],
pas_data[pas_index[2]]['data']), axis = 0)
print('Data for pas 1 and 2 shape: {}'.format(data_1_2.shape))
# Similarly, create a y_1_2 for the target vector
y_1_2 = y[np.where((y == 1) | (y == 2))]
print('y for pas 1 and 2 shape: {}'.format(y_1_2.shape))
#Our data_1_2 is a three-dimensional array. Our strategy will be to
collapse our two last dimensions (sensors and time) into one dimension,
while keeping the first dimension as it is (repetitions). Use np.reshape to
create a variable X_1_2 that fulfils these criteria.
X_1_2 = np.reshape(data_1_2, (data_1_2.shape[0], data_1_2.shape[1] *
data_1_2.shape[2]))
print('Reshaped data for pas 1 and 2 shape: {}'.format(X_1_2.shape))
#  and scale X_1_2
sc = StandardScaler()
sc.fit(X_1_2)
X_1_2_std = sc.transform(X_1_2)
# Do a standard LogisticRegression - make sure there is no penalty applied
lr = LogisticRegression(penalty='none', solver='lbfgs', multi_class='ovr',
n_jobs=-1)
lr.fit(X_1_2_std, y_1_2)
# we get a score of 1.0...that'd definitely an overfit, right? haha
print('Logistic regression score with no penalty:
{}'.format(lr.score(X_1_2_std, y_1_2)))
#found classes
print('Classes found in regression: {}'.format(lr.classes_))
# total coefs
coefs_orig = lr.coef_
coefs = len(lr.coef_[0])
print('Total coefs: {}'.format(coefs))
# non-zero coefs
coefs_nonzero = len(lr.coef_[0].nonzero()[0])
print('Nonzero coefs: {}'.format(coefs_nonzero))
# zero coefs
coefs_zero = coefs - coefs_nonzero
print('Zero coefs: {}'.format(coefs_zero))
# apply l1 penalty
lr = LogisticRegression(penalty='l1', solver='liblinear',
multi_class='ovr')
lr.fit(X_1_2_std, y_1_2)
```

```
print('Logistic regression score with l1 penalty:
{}'.format(lr.score(X_1_2_std, y_1_2)))
# default_threshold = np.get_printoptions()['threshold']
# np.set_printoptions(threshold=sys.maxsize)
# print(lr.coef_)
# np.set_printoptions(threshold=default_threshold)
#found classes
print('Classes found in regression: {}'.format(lr.classes_))
# total coefs
print('Total coefs: {}'.format(lr.coef_.shape))
coefs_orig = lr.coef_[0]
coefs = len(lr.coef_[0])
print('Total coefs: {}'.format(coefs))
# non-zero coefs
coefs_nonzero = len(lr.coef_[0].nonzero()[0])
print('Nonzero coefs: {}'.format(coefs_nonzero))
# zero coefs
coefs_zero = coefs - coefs_nonzero
print('Zero coefs: {}'.format(coefs_zero))
# part 2.1.3: something is wrong with the above code because there is only
one non-zero coefficient.

cov_nz_index = lr.coef_[0].nonzero()[0]
X_reduced = X_1_2_std[:,cov_nz_index]

cmat = X_reduced.T @ X_reduced

sns.heatmap(cmat, cmap='icefire', center = 0)
plt.show()
```

```
Data for pas 1 and 2 shape: (214, 102, 251)
y for pas 1 and 2 shape: (214,)
Reshaped data for pas 1 and 2 shape: (214, 25602)
Logistic regression score with no penalty: 1.0
Classes found in regression: [1 2]
Total coefs: 25602
Nonzero coefs: 25602
Zero coefs: 0
Logistic regression score with l1 penalty: 1.0
Classes found in regression: [1 2]
Total coefs: (1, 25602)
Total coefs: 25602
Nonzero coefs: 288
Zero coefs: 25314
```



2.1.v Are we overfitting? Besides the score, what would make you suspect that we are overfitting?

As we get a score of 1.0, it seems like an overfitting situation – besides the score of 1.0 we also suspect overfitting based on the lack of model-penalties.

2.1.vi Now apply the L1 penalty instead - how many of the coefficients (.coef_) are non-zero after this?

We found there to be 215 non-zero coefficients.

2.1.vii Plot the covariance of the features using plt.imshow. Compared to the plot from 1.1.iii, do we see less covariance?

It could seem as there is less covariance, it is however super difficult to assess this from these complex plots still. <3

Exercise 2.2 cross validation (VK)

In [ ]:

```
def equalize_targets_class(data, y):
    np.random.seed(7)
    targets = np.unique(y) ## find the number of targets
    if len(targets) < 2:
        raise NameError("at least 2 targets required")
    counts = list()
    indices = list()
    for target in targets:
        counts.append(np.sum(y == target)) ## find the number of each
target
        indices.append(np.where(y == target)[0]) ## find their indices
    min_count = np.min(counts)
    # randomly choose trials
    choices = []
    for i in range(len(targets)):
        choices.append(np.random.choice(indices[i], size=min_count,
replace=False))

    # create the new data sets
    new_indices = np.concatenate(tuple(choices))
    new_y = y[new_indices]
    new_data = data[new_indices, :, :]

    return new_data, new_y
```

2.2.iii Stratified 5-fold cross validation logistic model (KV)

In [ ]:

```
X_1_2_equal, y_1_2_equal = equalize_targets_class(data_1_2, y_1_2)
X_1_2_equal = np.reshape(X_1_2_equal, (X_1_2_equal.shape[0],
X_1_2_equal.shape[1] * X_1_2_equal.shape[2]))
sc.fit(X_1_2_equal)
X_1_2_equal_std = sc.transform(X_1_2_equal)
cv = StratifiedKFold(n_splits=5)
lr = LogisticRegression(penalty='none', solver='lbfgs', multi_class='ovr',
n_jobs=-1) # no regularisation
print('Getting cross validation score (Stratified 5-fold)...')
scores = cross_val_score(lr, X_1_2_equal_std, y_1_2_equal, cv=cv)
print('Cross validation score (Stratified 5-fold):
{}'.format(np.mean(scores)))
```

```
Getting cross validation score (Stratified 5-fold)...
Cross validation score (Stratified 5-fold): 0.4746153846153846
```

2.2.iv Stratified 5-fold cross validation logistic model with l2 (LR)

In [ ]:

```
best_score = 0.0
for c in [1e5, 1e1, 1e-5]:
  print('Getting cross validation score (Stratified 5-fold, C=
{})...'.format(c))
  lr = LogisticRegression(penalty='l2', solver='lbfgs', multi_class='ovr',
C=c, n_jobs=-1)
  scores = cross_val_score(lr, X_1_2_equal_std, y_1_2_equal, cv=cv)
  score = np.mean(scores)
  print('Cross validation score (Stratified 5-fold, C={}): {}'.format(c,
score))
  if score >= best_score:
    best_score = score
    best_c = c
print('Best C: {}'.format(best_c))
```

```
Getting cross validation score (Stratified 5-fold, C=100000.0)...
Cross validation score (Stratified 5-fold, C=100000.0): 0.4746153846153846
Getting cross validation score (Stratified 5-fold, C=10.0)...
Cross validation score (Stratified 5-fold, C=10.0): 0.4746153846153846
Getting cross validation score (Stratified 5-fold, C=1e-05)...
Cross validation score (Stratified 5-fold, C=1e-05): 0.45487179487179485
Best C: 10.0
```

2.2.iv In the best-scoring of these models, how many more/fewer predictions are correct (on average)?

It seems as if C=1e-05 gives the worst score at 0.455 and C=10.0 gives the best score at 0.475. The difference between them is thus 0.02. It is however surprising that C=1e5 seems to give almost the exact same number as C=1e1. This indicates that reularization is not making a large difference for these data.

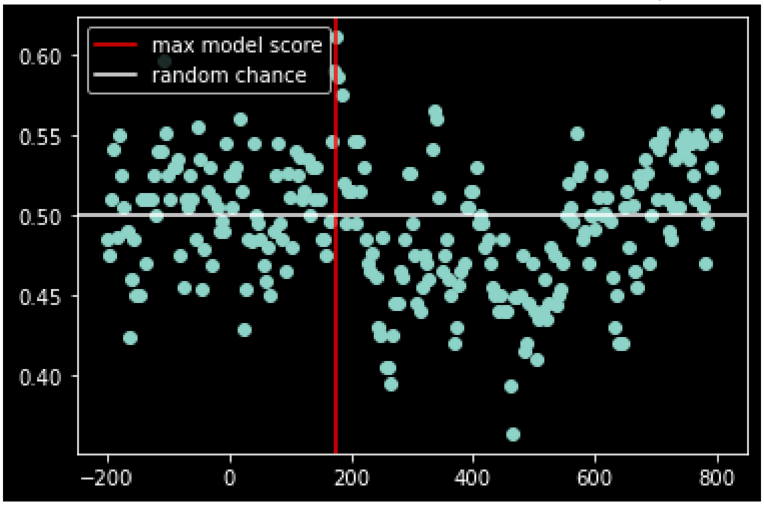
2.2.v now fit a model for each time sample using best c (EH)

In [ ]:

```
# 2.2.v
scores = []
lr = LogisticRegression(penalty='l2', solver='newton-cg',
multi_class='ovr', C=best_c, n_jobs=-1)
X_1_2_equal, y_1_2_equal = equalize_targets_class(data_1_2, y_1_2)
print('Fitting l2 model (c={}) for time sample: '.format(best_c), end='',
flush=True)
for time_ofs in range(data_1_2.shape[2]):
  X_1_2_equal_time = X_1_2_equal[:, :, time_ofs].squeeze()
  print('{}'.format(time_ofs), end='..', flush=True)
  # print('Shape of X: {}'.format(X_1_2_equal_time.shape))
  sc.fit(X_1_2_equal_time)
  X_1_2_equal_std = sc.transform(X_1_2_equal_time)
  # print('Getting Cross validation score (Stratified 5-fold, C={}, time=
{}ms)'.format(best_c, times[time_ofs]))
  time_scores = cross_val_score(lr, X_1_2_equal_std, y_1_2_equal, cv=cv)
  score = np.mean(time_scores)
  # print('Cross validation score (Stratified 5-fold, C={}, time={}ms):
{}'.format(best_c, times[time_ofs], score))
  scores.append(score)
print('\nCompleted fitting time-based models')
scores = np.array(scores)
max_score = np.unravel_index(np.argmax(scores), scores.shape)
print('Best score: {} at timestamp {}ms'.format(scores[max_score],
times[max_score[0]]))
plt.scatter(times, scores)
```

```
plt.axvline(x = times[max_score[0]], color='r', label='max model score')
plt.axhline(y = 0.5, label = 'random chance', color='w')
plt.legend(loc="upper left")
plt.show()
```

```
Fitting l2 model (c=10.0) for time sample: 0..1..2..3..4..5..6..7..8..9..10..
11..12..13..14..15..16..17..18..19..20..21..22..23..24..25..26..27..28..29..3
0..31..32..33..34..35..36..37..38..39..40..41..42..43..44..45..46..47..48..4
9..50..51..52..53..54..55..56..57..58..59..60..61..62..63..64..65..66..67..6
8..69..70..71..72..73..74..75..76..77..78..79..80..81..82..83..84..85..86..8
7..88..89..90..91..92..93..94..95..96..97..98..99..100..101..102..103..104..1
05..106..107..108..109..110..111..112..113..114..115..116..117..118..119..12
0..121..122..123..124..125..126..127..128..129..130..131..132..133..134..13
5..136..137..138..139..140..141..142..143..144..145..146..147..148..149..15
0..151..152..153..154..155..156..157..158..159..160..161..162..163..164..16
5..166..167..168..169..170..171..172..173..174..175..176..177..178..179..18
0..181..182..183..184..185..186..187..188..189..190..191..192..193..194..19
5..196..197..198..199..200..201..202..203..204..205..206..207..208..209..21
0..211..212..213..214..215..216..217..218..219..220..221..222..223..224..22
5..226..227..228..229..230..231..232..233..234..235..236..237..238..239..24
0..241..242..243..244..245..246..247..248..249..250..
Completed fitting time-based models
Best score: 0.611025641025641 at timestamp 176ms
```



2.2.v What are the time points where classification is best? Make a plot with time on the x-axis and classification score on the y-axis with a horizontal line at the chance level (what is the chance level for this analysis?)

The best score 61.1% is at timestamp = 176 ms. Chance level should be at 50% as we are predicting PAS 1 and PAS 2

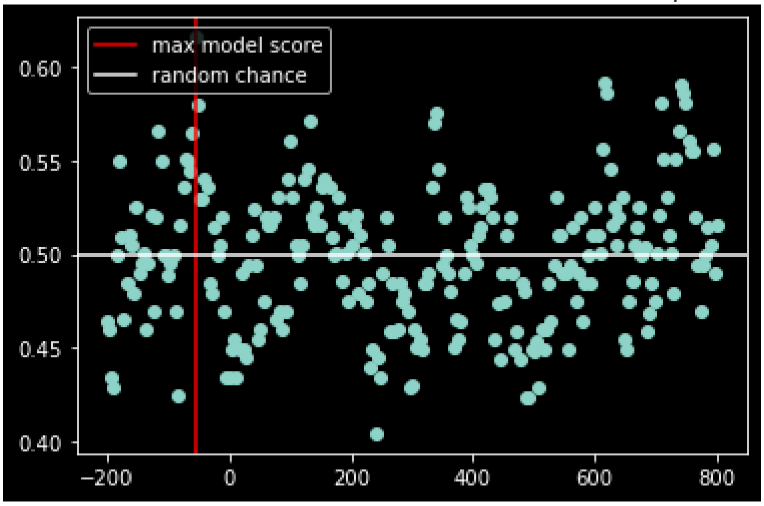
2.2.vi now fit a model for each time sample using c=1e-1 and l1 (VK)

In [ ]:

```
scores = []
lr = LogisticRegression(penalty='l1', solver='liblinear',
multi_class='ovr', C=1e-1)
X_1_2_equal, y_1_2_equal = equalize_targets_class(data_1_2, y_1_2)
print('Fitting l1 model (c=1e-1) for time sample: ', end='', flush=True)
for time_ofs in range(data_1_2.shape[2]):
  X_1_2_equal_time = X_1_2_equal[:, :, time_ofs].squeeze()
  print('{}'.format(time_ofs), end='..', flush=True)
  # print('Shape of X: {}'.format(X_1_2_equal_time.shape))
  sc.fit(X_1_2_equal_time)
  X_1_2_equal_std = sc.transform(X_1_2_equal_time)
  # print('Getting Cross validation score (Stratified 5-fold, C={}, time=
{}ms)'.format(best_c, times[time_ofs]))
  time_scores = cross_val_score(lr, X_1_2_equal_std, y_1_2_equal, cv=cv)
```

```
    score = np.mean(time_scores)
    # print('Cross validation score (Stratified 5-fold, C={}, time={}ms):
{}'.format(best_c, times[time_ofs], score))
    scores.append(score)
print('\nCompleted fitting time-based models')
scores = np.array(scores)
max_score = np.unravel_index(np.argmax(scores), scores.shape)
print('Best score: {} at timestamp {}ms'.format(scores[max_score],
times[max_score[0]]))
plt.scatter(times, scores)
plt.axvline(x = times[max_score[0]], color='r', label='max model score')
plt.axhline(y = 0.5, label = 'random chance', color='w')
plt.legend(loc="upper left")
plt.show()
```

```
Fitting l1 model (c=1e-1) for time sample: 0..1..2..3..4..5..6..7..8..9..10..
11..12..13..14..15..16..17..18..19..20..21..22..23..24..25..26..27..28..29..3
0..31..32..33..34..35..36..37..38..39..40..41..42..43..44..45..46..47..48..4
9..50..51..52..53..54..55..56..57..58..59..60..61..62..63..64..65..66..67..6
8..69..70..71..72..73..74..75..76..77..78..79..80..81..82..83..84..85..86..8
7..88..89..90..91..92..93..94..95..96..97..98..99..100..101..102..103..104..1
05..106..107..108..109..110..111..112..113..114..115..116..117..118..119..12
0..121..122..123..124..125..126..127..128..129..130..131..132..133..134..13
5..136..137..138..139..140..141..142..143..144..145..146..147..148..149..15
0..151..152..153..154..155..156..157..158..159..160..161..162..163..164..16
5..166..167..168..169..170..171..172..173..174..175..176..177..178..179..18
0..181..182..183..184..185..186..187..188..189..190..191..192..193..194..19
5..196..197..198..199..200..201..202..203..204..205..206..207..208..209..21
0..211..212..213..214..215..216..217..218..219..220..221..222..223..224..22
5..226..227..228..229..230..231..232..233..234..235..236..237..238..239..24
0..241..242..243..244..245..246..247..248..249..250..
Completed fitting time-based models
Best score: 0.6161538461538462 at timestamp -56ms
```



2.2.vi what are the time points when classification is best? (make a plot)?

classification accuracy was best (61.62%) at time point -56ms.

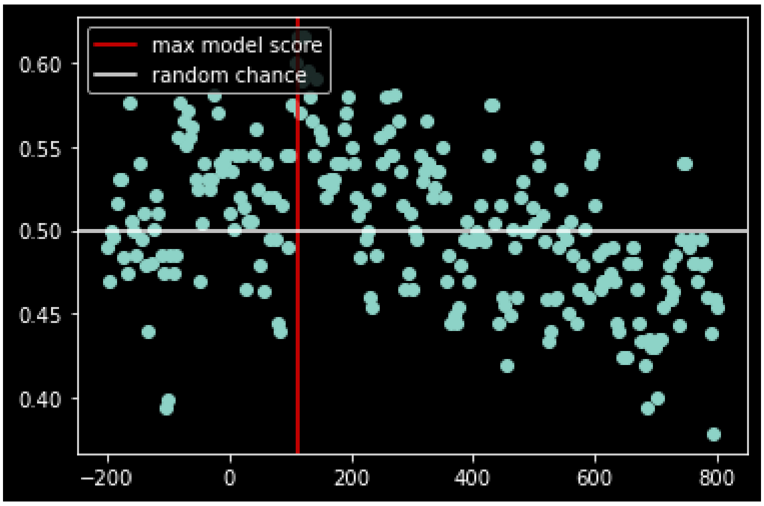
2.2.vii fit again but for PAS 1 and 4 (KV)

In [ ]:

```
data_1_4 = np.concatenate((pas_data[pas_index[1]]['data'],
pas_data[pas_index[4]]['data']), axis = 0)
print('Data for pas 1 and  4 shape: {}'.format(data_1_4.shape))
y_1_4 = y[np.where((y == 1) | (y == 2))]
print('y for pas 1 and 4 shape: {}'.format(y_1_4.shape))
#Our data_1_2 is a three-dimensional array. Our strategy will be to
collapse our two last dimensions (sensors and time) into one dimension,
```

```
while keeping the first dimension as it is (repetitions). Use np.reshape to
create a variable X_1_2 that fulfils these criteria.
X_1_4 = np.reshape(data_1_4, (data_1_4.shape[0], data_1_4.shape[1] *
data_1_4.shape[2]))
scores = []
lr = LogisticRegression(penalty='l1', solver='liblinear',
multi_class='ovr', C=1e-1)
X_1_4_equal, y_1_4_equal = equalize_targets_class(data_1_4, y_1_4)
print('Fitting l1 model (c=1e-1) for time sample: ', end='', flush=True)
for time_ofs in range(data_1_4.shape[2]):
  X_1_4_equal_time = X_1_4_equal[:, :, time_ofs].squeeze()
  print('{}'.format(time_ofs), end='..', flush=True)
  # print('Shape of X: {}'.format(X_1_4_equal_time.shape))
  sc.fit(X_1_4_equal_time)
  X_1_4_equal_std = sc.transform(X_1_4_equal_time)
  # print('Getting Cross validation score (Stratified 5-fold, C={}, time=
{}ms)'.format(best_c, times[time_ofs]))
  time_scores = cross_val_score(lr, X_1_4_equal_std, y_1_4_equal, cv=cv)
  score = np.mean(time_scores)
  # print('Cross validation score (Stratified 5-fold, C={}, time={}ms):
{}'.format(best_c, times[time_ofs], score))
  scores.append(score)
print('\nCompleted fitting time-based models')
scores = np.array(scores)
max_score = np.unravel_index(np.argmax(scores), scores.shape)
print('Best score: {} at timestamp {}ms'.format(scores[max_score],
times[max_score[0]]))
plt.scatter(times, scores)
plt.axvline(x = times[max_score[0]], color='r', label='max model score')
plt.axhline(y = 0.5, label = 'random chance', color='w')
plt.legend(loc="upper left")
plt.show()
```

```
Data for pas 1 and  4 shape: (359, 102, 251)
y for pas 1 and 4 shape: (214,)
Fitting l1 model (c=1e-1) for time sample: 0..1..2..3..4..5..6..7..8..9..10..
11..12..13..14..15..16..17..18..19..20..21..22..23..24..25..26..27..28..29..3
0..31..32..33..34..35..36..37..38..39..40..41..42..43..44..45..46..47..48..4
9..50..51..52..53..54..55..56..57..58..59..60..61..62..63..64..65..66..67..6
8..69..70..71..72..73..74..75..76..77..78..79..80..81..82..83..84..85..86..8
7..88..89..90..91..92..93..94..95..96..97..98..99..100..101..102..103..104..1
05..106..107..108..109..110..111..112..113..114..115..116..117..118..119..12
0..121..122..123..124..125..126..127..128..129..130..131..132..133..134..13
5..136..137..138..139..140..141..142..143..144..145..146..147..148..149..15
0..151..152..153..154..155..156..157..158..159..160..161..162..163..164..16
5..166..167..168..169..170..171..172..173..174..175..176..177..178..179..18
0..181..182..183..184..185..186..187..188..189..190..191..192..193..194..19
5..196..197..198..199..200..201..202..203..204..205..206..207..208..209..21
0..211..212..213..214..215..216..217..218..219..220..221..222..223..224..22
5..226..227..228..229..230..231..232..233..234..235..236..237..238..239..24
0..241..242..243..244..245..246..247..248..249..250..
Completed fitting time-based models
Best score: 0.6157692307692308 at timestamp 112ms
```



2.2.vii What are the time points when classification is best? ... what is the chance level for this analysis?

Best classification (61.58%) is at time point 112ms. Chance level for this analysis should be 50% as we are working with PAS 1 and PAS 4.

2.3 Is pairwise classification of subjective experience possible? Any surprises in the classification accuracies, i.e. how does the classification score fore PAS 1 vs 4 compare to the classification score for PAS 1 vs 2?

As our scores is all above chance level (however not a lot) it seems to be possible to do pairwise classification. It is quite surprising that the accuracy level of PAS 1 vs 2 and PAS 1 vs 4 are almost identical, though at very different time points.

EXERCISE 3 - Do a Support Vector Machine Classification on all four PAS-ratings

3.1 Support Vector Machine Classification (LR)

3.1.i First equalize the number of targets using the function associated with each PAS-rating using the function associated with Exercise 3.1.i

In [ ]:

```
X_equal, y_equal = equalize_targets_class(data, y)
print('X shape: {}'.format(X_equal.shape))
print('y shape: {}'.format(y_equal.shape))

X_equal_f = np.reshape(X_equal, (X_equal.shape[0], X_equal.shape[1] *
X_equal.shape[2]))

sc.fit(X_equal_f)
X_equal_std = sc.transform(X_equal_f)

print('X reshaped shape: {}'.format(X_equal_std.shape))
```

```
X shape: (396, 102, 251)
y shape: (396,)
X reshaped shape: (396, 25602)
```

3.1.ii Linear kernel classifier (LR)

In [ ]:

```
cv = StratifiedKFold()
svm = SVC(kernel='linear')
scores_svm = cross_val_score(svm, X_equal_std, y_equal, cv=cv)
print(np.mean(scores_svm))
```

```
0.2928164556962025
```

3.1.ii Radial basis (EH)

In [ ]:
```
cv = StratifiedKFold()
svm = SVC(kernel='rbf')
scores_svm = cross_val_score(svm, X_equal_std, y_equal, cv=cv)
print(np.mean(scores_svm))
```

```
0.3333544303797468
```

3.1.ii) Which one is better predicting the category?

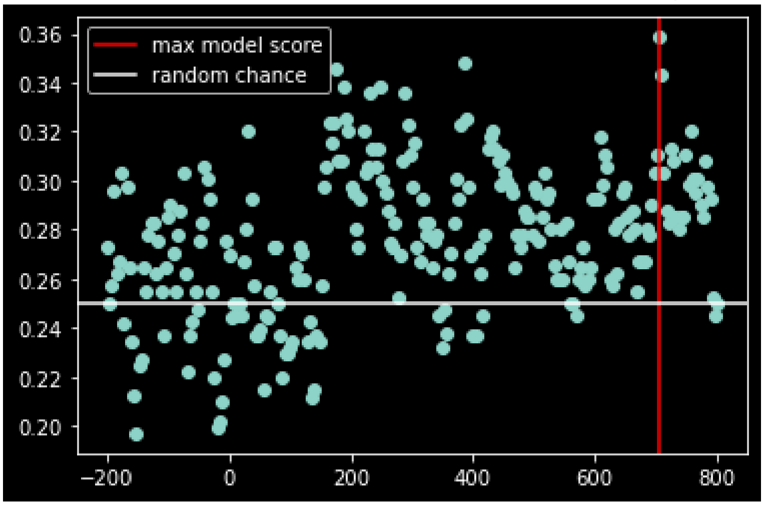
Based on the cross-validation results, the linear kernel classifier achieved 29.9% accuracy, and the radial basis kernel classifier achieved 33.4% accuracy, so for these data the radial basis kernel classifier is the best predictor.

3.3.iv Sample-by-sample analysis (VK)

In [ ]:
```
# radial basis score better than linear
cv = StratifiedKFold()
svm = SVC(kernel='rbf')

scores = []
print(X_equal.shape)
print('Fitting rbv svm model time sample: ', end='', flush=True)
for time_ofs in range(X_equal.shape[2]):
  X_equal_time = X_equal[:, :, time_ofs].squeeze()
  print('{}'.format(time_ofs), end='..', flush=True)
  # print('Shape of X: {}'.format(X_1_2_equal_time.shape))
  sc.fit(X_equal_time)
  X_equal_std = sc.transform(X_equal_time)
  # print('Getting Cross validation score (Stratified 5-fold, C={}, time=
{}ms)'.format(best_c, times[time_ofs]))
  time_scores = cross_val_score(svm, X_equal_std, y_equal, cv=cv)
  score = np.mean(time_scores)
  # print('Cross validation score (Stratified 5-fold, C={}, time={}ms):
{}'.format(best_c, times[time_ofs], score))
  scores.append(score)
print('\nCompleted fitting time-based models')
scores = np.array(scores)
max_score = np.unravel_index(np.argmax(scores), scores.shape)
print('Best score: {} at timestamp {}ms'.format(scores[max_score],
times[max_score[0]]))
plt.scatter(times, scores)
# random chance is 1/4
plt.axvline(x = times[max_score[0]], color='r', label='max model score')
plt.axhline(y = 0.25, label = 'random chance', color='w')
plt.legend(loc="upper left")
plt.show()
```

```
(396, 102, 251)
Fitting rbv svm model time sample: 0..1..2..3..4..5..6..7..8..9..10..11..12..
13..14..15..16..17..18..19..20..21..22..23..24..25..26..27..28..29..30..31..3
2..33..34..35..36..37..38..39..40..41..42..43..44..45..46..47..48..49..50..5
1..52..53..54..55..56..57..58..59..60..61..62..63..64..65..66..67..68..69..7
```

```
0..71..72..73..74..75..76..77..78..79..80..81..82..83..84..85..86..87..88..8
9..90..91..92..93..94..95..96..97..98..99..100..101..102..103..104..105..10
6..107..108..109..110..111..112..113..114..115..116..117..118..119..120..12
1..122..123..124..125..126..127..128..129..130..131..132..133..134..135..13
6..137..138..139..140..141..142..143..144..145..146..147..148..149..150..15
1..152..153..154..155..156..157..158..159..160..161..162..163..164..165..16
6..167..168..169..170..171..172..173..174..175..176..177..178..179..180..18
1..182..183..184..185..186..187..188..189..190..191..192..193..194..195..19
6..197..198..199..200..201..202..203..204..205..206..207..208..209..210..21
1..212..213..214..215..216..217..218..219..220..221..222..223..224..225..22
6..227..228..229..230..231..232..233..234..235..236..237..238..239..240..24
1..242..243..244..245..246..247..248..249..250..
Completed fitting time-based models
Best score: 0.3586075949367088 at timestamp 704ms
```



3.1.iv) Is classification of subjective experience possible at around 200-250 ms?

It seems that the model would perform above random chance at classification of subjective experience for 200-250ms although the accuracy is still not high, ranging from around 0.27-0.34.

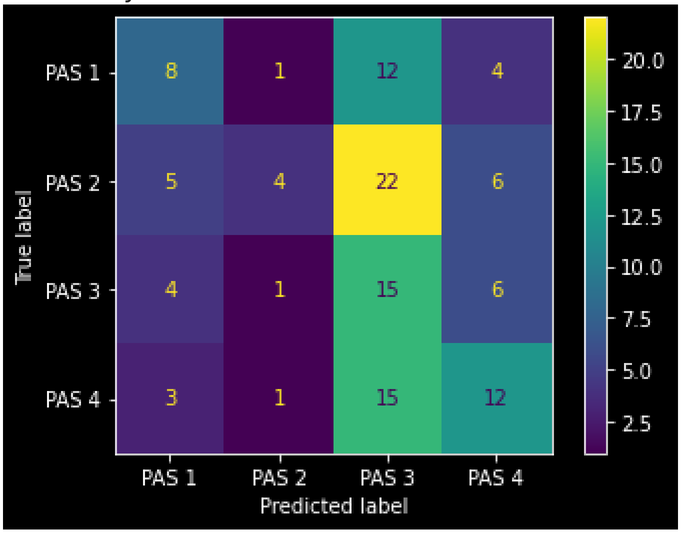
Exercise 3.2 Test/train (KV)

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X_equal, y_equal,
test_size=0.3)
print(X_train.shape)
print(y_train.shape)
# assignment says: This time your features are the number of sensors
multiplied by the number of samples.
# So, we need to reshape the data to be (n_samples, n_features)
X_train_f = np.reshape(X_train, (X_train.shape[0], X_train.shape[1] *
X_train.shape[2]))
X_test_f = np.reshape(X_test, (X_test.shape[0], X_test.shape[1] *
X_test.shape[2]))

sc.fit(X_train_f)
X_train_f_std = sc.transform(X_train_f)
X_test_f_std = sc.transform(X_test_f)

svm = SVC(kernel='rbf')

svm.fit(X_train_f_std, y_train)
predictions = svm.predict(X_test_f_std)
print('Accuracy: {}'.format(accuracy_score(y_test, predictions)))
# confusion matrix
# print(confusion_matrix(y_test, predictions))
ConfusionMatrixDisplay.from_predictions(y_test, predictions,
```

```
display_labels=['PAS 1', 'PAS 2', 'PAS 3', 'PAS 4'])
plt.show()
```

```
(277, 102, 251)
(277,)
Accuracy: 0.3277310924369748
```



3.2.iii) Based on the confusion matrix, describe how ratings are misclassified and if that makes sense given that ratings should measure the strength/quality of the subjective experience. Is the classifier biased towards specific ratings?

The confusion matrix shows a bias towards predicting PAS 3, which can be seen from the PAS 3 predicted label on the x axis, this should not come from PAS 3 being the most frequent rating, because the samples were stratified. Of the predicted labels classified as PAS 1, 2 and 4, the prediction was correct most frequently, which fits with subjects indicating that they either not aware of the stimulus (PAS 1) or did not doubt their experience of the stimulus (PAS 4). The PAS 3 label was most frequently misclassified, especially when the true label was PAS 2, but both 2 and 3 ratings that relate to uncertainty and ambiguity.

Portfolio Assignment 4

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

Student numbers

- Emma Rose Hahn: 202004249
- Kristian Nøhr Villebro: 202005289
- Luke Ring: 202009983
- Viara Krasteva: 202005673

In [ ]:

```
# methods3 conda environment
# imports
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:

```
# common functions
def equalize_targets(data, y):
    np.random.seed(7)
    targets = np.unique(y)
    counts = list()
    indices = list()
    for target in targets:
        counts.append(np.sum(y == target))
        indices.append(np.where(y == target)[0])
    min_count = np.min(counts)
    first_choice = np.random.choice(indices[0], size=min_count,
replace=False)
    second_choice = np.random.choice(indices[1], size=min_count,
replace=False)
    third_choice = np.random.choice(indices[2], size=min_count,
replace=False)
    fourth_choice = np.random.choice(indices[3], size=min_count,
replace=False)

    new_indices = np.concatenate((first_choice, second_choice,
                                 third_choice, fourth_choice))
    new_y = y[new_indices]
    new_data = data[new_indices, :, :]

    return new_data, new_y
```

Exercise 1 - Use principal component analysis to improve the classification of subjective experience

1.1 - Create a covariance matrix, find the eigenvectors and the eigenvalues (EH)

In [ ]:

```
data = np.load('megmag_data.npy')
pas_data = np.load('pas_vector.npy')
```

1.1.ii) Equalize the number of targets in y and data using equalize_targets

In [ ]:

```
data, y = equalize_targets(data, pas_data)
```

1.1.iii) times and 1.1.iii) reduce to two dimensions only using 248ms

In [ ]:

```
times=np.arange(-200, 804, 4)
data_reduced = data[:, :, np.where(times == 248)[0][0]]
```

1.1.iv) Scale the data using StandardScaler

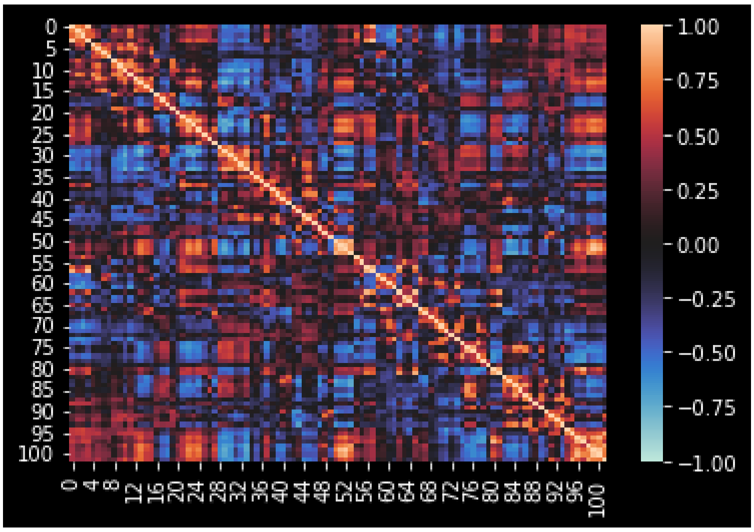
In [ ]:

```
sc = StandardScaler()
data_scaled = sc.fit_transform(data_reduced, y)
```

1.1.v) Calculate the sample covariance matrix using np.cov

In [ ]:

```
cov_matrix = np.cov(data_scaled, rowvar=False)
# plot using seaborn heatmap
sns.heatmap(cov_matrix, cmap='icefire', vmin=-1, vmax=1)
plt.show()
```



1.1.vi) What does the off-diagonal activation imply about the independence of the signals measured by the 102 sensors? (KV)

We see that the off-diagonal activation implies that the signals measured by the sensors are not independent. This makes sense as activity in the brain can involve multiple regions, and we would expect to see some correlation and covariance between sensors.

1.1.vii Run np.linalg.matrix_rank on the covariance matrix - what integer value do you get? (we'll use this later)

In [ ]:

```
rank = np.linalg.matrix_rank(cov_matrix)
print("Matrix rank: {}".format(rank))
```

```
Matrix rank: 97
```

1.1.viii) Find the eigenvalues and eigenvectors of the covariance matrix using np.linalg.eig

In [ ]:

```
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)
# Use np.real to retrieve only the real parts of the eigenvalues and
eigenvectors
eigenvalues = np.abs(np.real(eigenvalues))
eigenvectors = np.real(eigenvectors)
```

1.2 - Create the weighting matrix W and the projected data, Z

1.2.i to 1.2.iii) sort the eigenvectors and eigenvalues according to the absolute values of the eigenvalues

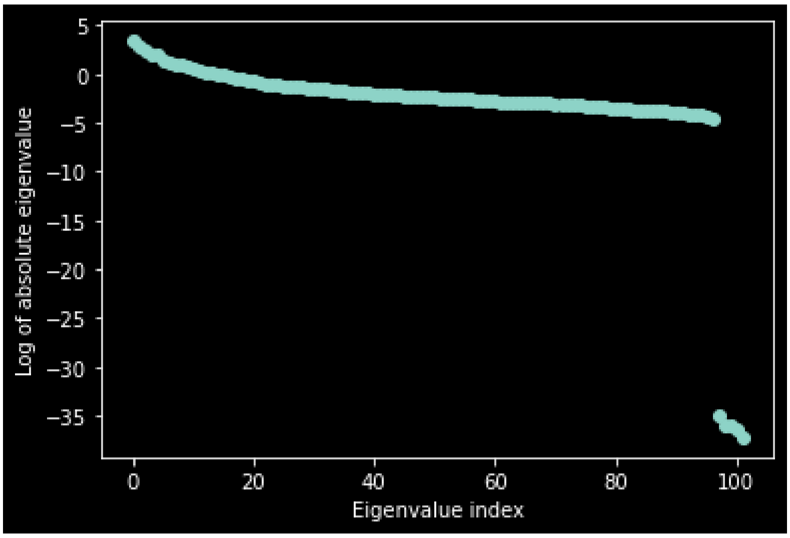
In [ ]:

```
# get the sorted indices of absolute values in reverse order
sorted_indices = np.argsort(eigenvalues)[::-1]
# sorted eigenvalues
eigenvalues = eigenvalues[sorted_indices]
# sorted eigenvectors
eigenvectors = eigenvectors[:, sorted_indices]
```

1.2.iv) Plot the log of the eigenvalues

In [ ]:

```
plt.plot(np.log(eigenvalues), 'o')
plt.xlabel('Eigenvalue index')
plt.ylabel('Log of absolute eigenvalue')
plt.show()
```



1.2.iv) are there some values that stand out from the rest? (LR)

The last 5 of the sorted log eigenvalues are much smaller than the previous 97. This matches the result of our matrix rank test which gives values above a threshold.

1.2.v - 1.2.vi) eigenvectors are the weighting matrix, create projected data Z = W.X

In [ ]:

```
Z = data_scaled @ eigenvectors
X = Z @ eigenvectors.T

# check if calculations are correct
closeness_check = np.isclose(data_scaled, X)
```

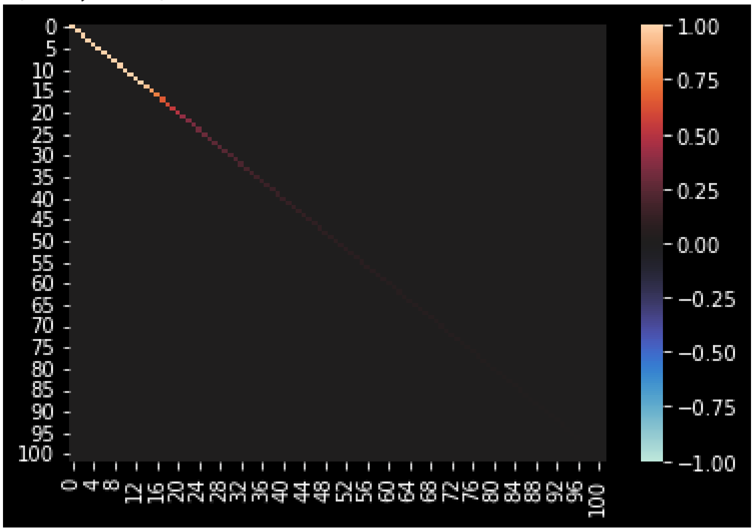
```
number_of_false_values = np.sum(closeness_check == False)
print("Number values that are not close:
{}".format(number_of_false_values))
```

```
Number values that are not close: 0
```

In [ ]:

```
# 1.2.vii) Create a new covariance matrix of the principal components
(n=102)
cov_matrix_pca = np.cov(Z, rowvar=False)
print(Z.shape)
# plot using seaborn heatmap
sns.heatmap(cov_matrix_pca, cmap='icefire', vmin=-1, vmax=1)
plt.show()
```

```
(396, 102)
```



1.2.vii What has happened to the off-diagonals and why? (LR)

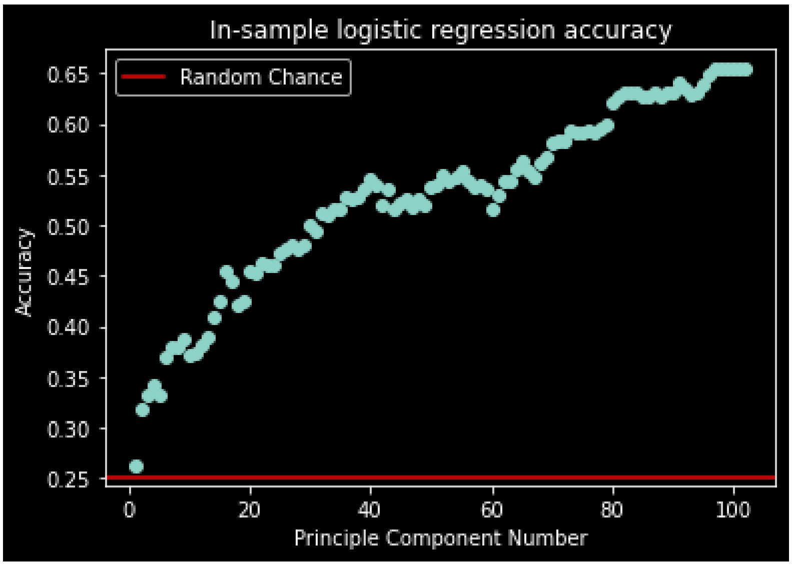
The weighting matrix is the orthogonal projection of the data onto the eigenvectors, this results in values that are not correlated with each other, giving a 0 value for the covariance matrix.

Exercise 2 - Use logistic regression with cross-validation to find the optimal number of principal components (VK)

2.1.i) run standard logistic regression (no regularization) based on Zdxk and y

In [ ]:

```
lr = LogisticRegression(fit_intercept=False, solver='newton-cg')
# fit 102 models based on: k=[1,2,...,101,102] and d=102. For each fit get
the classification accuracy, (.score), when applied to Z_d×k and y
scores = list()
d = Z.shape[1]
for k in range(1, d+1):
    Z_kxd = Z[:, :k]
    lr.fit(Z_kxd, y)
    scores.append(lr.score(Z_kxd, y))

plt.scatter(np.arange(1, d+1), scores)
plt.axhline(0.25, color='red', label='Random Chance')
plt.xlabel('Principle Component Number')
plt.ylabel('Accuracy')
plt.title('In-sample logistic regression accuracy')
plt.legend()
plt.show()
```



2.1.ii) what is the general trend and why is this so? (EH)

Based on the above plot we could say the general trend is that addming more principle components results in a higher classification accuracy. This is because more variance is accounted for, but it also could mean that the model is overfitting.

2.1.iii) In terms of classification accuracy, what is the effect of adding the five last components? Why do you think this is so? (KV)

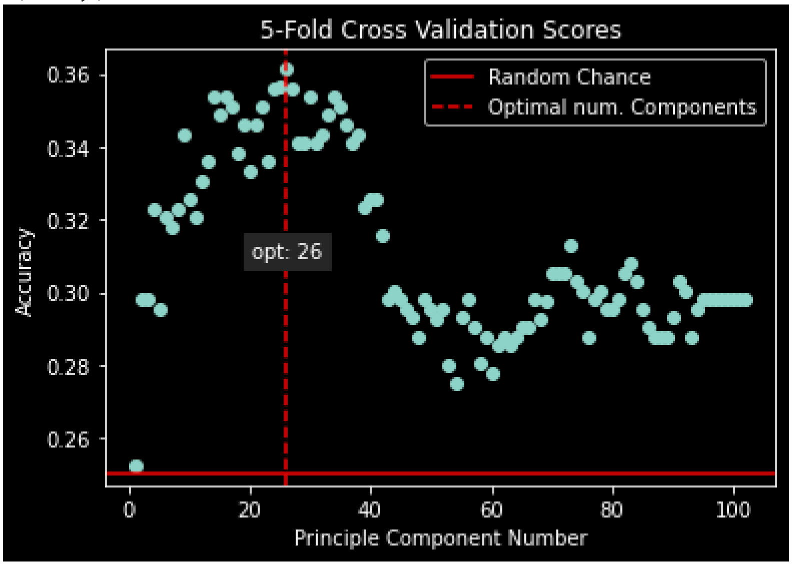
The last 5 components have a negligible effect, this is because they are the principle components that have the least variance, adding them to our models doesn't provide much additional information.

2.2) cross validate using cross_val_score and StratifiedKFold (LR)

In [ ]:

```
cv = StratifiedKFold()
scores = []
for k in range(1, d+1):
  Z_k = Z[:, :k]
  scores.append(cross_val_score(lr, Z_k, y, cv=cv))

scores = np.asarray(scores)
# get mean score for each k
mean_scores = np.mean(scores, axis=1)
# get index of max
max_index = np.argmax(mean_scores)
grand_mean = np.mean(mean_scores)

print(mean_scores.shape)
plt.scatter(np.arange(1, d+1), mean_scores)
plt.axhline(0.25, color='red', label='Random Chance')
plt.axvline(max_index+1, color='red', label='Optimal num. Components',
linestyle='--')
plt.text(max_index+1, grand_mean, 'opt: {}'.format(max_index+1),
horizontalalignment='center', verticalalignment='center',
backgroundcolor='#262626', color='white')
plt.xlabel('Principle Component Number')
plt.ylabel('Accuracy')
plt.title('5-Fold Cross Validation Scores')
plt.legend()
plt.show()
```

```
(102,)
```



2.2.ii) how is this plot different from the one in Exercise 2.1.ii? (VK)

This shows a different trend, rather than each additional component increasing the model accuracy, we see a peak at 26 components and after that a decline in accuracy which then levels out at around 70 components. Based on the first model we might assume that adding all the components would result in a higher accuracy, where as using cross-validation we see that the results of the previous model were likely due to overfitting.

2.2.iii) What is the number of principal components, $k_{max\_accuracy}$, that results in the greatest classification accuracy when cross-validated?

The peak model performance is 26 components, at around 36% accuracy.

In [ ]:

```
print("Maximum classification accuracy: {:.2%}, classifaction accuracy with full dataset: {:.2%}, improvement: {:.2%}".format(mean_scores[max_index], mean_scores[-1], (mean_scores[max_index]-mean_scores[-1])))
```

```
Maximum classification accuracy: 36.13%, classifaction accuracy with full dat
aset: 29.79%, improvement: 6.33%
```

2.2.iv) How many percentage points is the classification accuracy increased with relative to the to the full-dimensional, d, dataset? (EH)

using all of the components, the 5-fold cross-validation accuracy is increased by about 6.3%.

2.2.v) How do the analyses in Exercises 2.1 and 2.2 differ from one another? Make sure to comment on the differences in optimization criteria. (KV)

Both use a logistic regression model, the key difference here is that the first analysis is adding components and fitting the model to the entire dataset. 2.2 uses stratified 5-fold cross-validation which tries to ensure balanced classes across the folds, as well as training and testing on different subsets of the data. This is a more robust method of testing the model, as it is less prone to overfitting and while the classification accuracy may seem lower, the accuracy in 2.1 is likely overinflated.

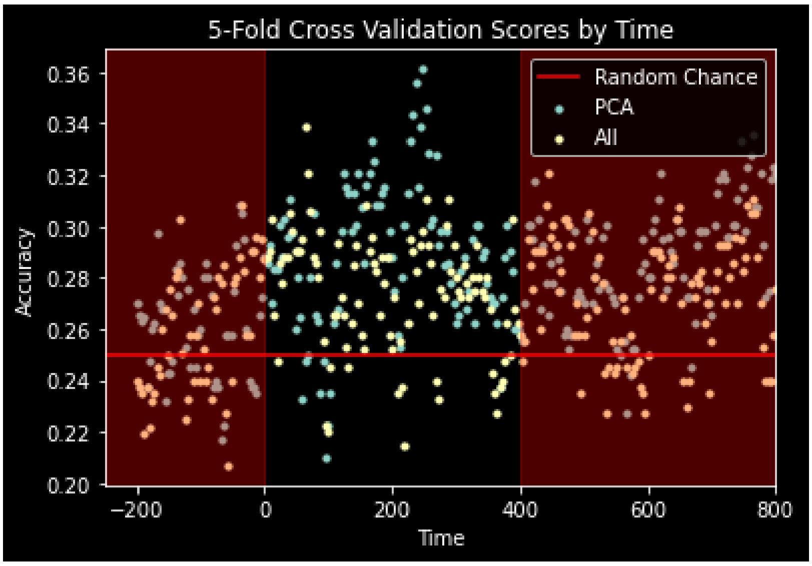
2.3.i) For each of the 251 time samples, use the same estimator and cross-validation as in Exercises 2.1.i and 2.2.i. (LR)

In [ ]:
```
scores_pca = []
scores_all = []
time_steps = len(times)
print("Running analyses for {} time samples from {}ms to
{}ms:".format(time_steps, times[0], times[-1]), flush=True)
for t in range(time_steps):
  print("{}ms".format(times[t]), end="", flush=True)
  # get data for current timestamp
  data_t = data[:, :, t]
  data_t_scaled = sc.transform(data_t)
  # use pca to reduce to k_max
  pca_kmax = PCA(n_components=max_index+1)
  pcs = pca_kmax.fit_transform(data_t_scaled)
  # calculate cross validation scores for reduced data
  scores_pca.append(cross_val_score(lr, pcs, y, cv=cv))
  print(".", end="", flush=True)
  # calculate cross validation scores for all data
  scores_all.append(cross_val_score(lr, data_t_scaled, y, cv=cv))
  print(".", end="", flush=True)
print("done\nPreparing plot...", flush=True)
```

```
Running analyses for 251 time samples from -200ms to 800ms:
-200ms..-196ms..-192ms..-188ms..-184ms..-180ms..-176ms..-172ms..-168ms..-164m
s..-160ms..-156ms..-152ms..-148ms..-144ms..-140ms..-136ms..-132ms..-128ms..-1
24ms..-120ms..-116ms..-112ms..-108ms..-104ms..-100ms..-96ms..-92ms..-88ms..-8
4ms..-80ms..-76ms..-72ms..-68ms..-64ms..-60ms..-56ms..-52ms..-48ms..-44ms..-4
0ms..-36ms..-32ms..-28ms..-24ms..-20ms..-16ms..-12ms..-8ms..-4ms..0ms..4ms..8
ms..12ms..16ms..20ms..24ms..28ms..32ms..36ms..40ms..44ms..48ms..52ms..56ms..6
0ms..64ms..68ms..72ms..76ms..80ms..84ms..88ms..92ms..96ms..100ms..104ms..108m
s..112ms..116ms..120ms..124ms..128ms..132ms..136ms..140ms..144ms..148ms..152m
s..156ms..160ms..164ms..168ms..172ms..176ms..180ms..184ms..188ms..192ms..196m
s..200ms..204ms..208ms..212ms..216ms..220ms..224ms..228ms..232ms..236ms..240m
s..244ms..248ms..252ms..256ms..260ms..264ms..268ms..272ms..276ms..280ms..284m
s..288ms..292ms..296ms..300ms..304ms..308ms..312ms..316ms..320ms..324ms..328m
s..332ms..336ms..340ms..344ms..348ms..352ms..356ms..360ms..364ms..368ms..372m
s..376ms..380ms..384ms..388ms..392ms..396ms..400ms..404ms..408ms..412ms..416m
s..420ms..424ms..428ms..432ms..436ms..440ms..444ms..448ms..452ms..456ms..460m
s..464ms..468ms..472ms..476ms..480ms..484ms..488ms..492ms..496ms..500ms..504m
s..508ms..512ms..516ms..520ms..524ms..528ms..532ms..536ms..540ms..544ms..548m
s..552ms..556ms..560ms..564ms..568ms..572ms..576ms..580ms..584ms..588ms..592m
s..596ms..600ms..604ms..608ms..612ms..616ms..620ms..624ms..628ms..632ms..636m
s..640ms..644ms..648ms..652ms..656ms..660ms..664ms..668ms..672ms..676ms..680m
s..684ms..688ms..692ms..696ms..700ms..704ms..708ms..712ms..716ms..720ms..724m
s..728ms..732ms..736ms..740ms..744ms..748ms..752ms..756ms..760ms..764ms..768m
s..772ms..776ms..780ms..784ms..788ms..792ms..796ms..800ms..done
Preparing plot...
```

In [ ]:
```
# plot scores by time (VK)
mean_pca_scores = np.mean(scores_pca, axis=1)
mean_all_scores = np.mean(scores_all, axis=1)

fig, ax = plt.subplots()
ax.axhline(0.25, color='red', label='Random Chance')
ax.scatter(times, mean_pca_scores, label='PCA', s=10)
ax.scatter(times, mean_all_scores, label='All', s=10)
ax.axvspan(-250, 0, color="red", alpha=0.3)
ax.axvspan(400, 800, color="red", alpha=0.3)
plt.xlim([-250, 800])
plt.xlabel('Time')
plt.ylabel('Accuracy')
plt.title('5-Fold Cross Validation Scores by Time')
```

```
plt.legend(loc='upper right')
plt.show()
```



2.3.iii) Describe the differences between the two analyses - focus on the time interval between 0 ms and 400 ms - describe in your own words why the logistic regression performs better on the PCA-reduced dataset around the peak magnetic activity (EH)

The PCA reduced dataset performance tends to be higher than the full dataset, especially around peak activity. The full dataset contains the signal from all the sensors, but some of these may be less relevant to the response being investigated thereby introducing noise into the model. As hypothesized in 2.2.v, the overall accuracy of the full dataset was inflated due to overfitting which reduced its ability to be generalized to other time intervals. Unsurprisingly, as the PCA model underwent more robust validation, the accuracy at the previously modelled timestamp (248ms) remained the same (~36%), and seems to be able to generalize to time intervals around the peak activity, where the signal is likely similar.