36-463/663: Multilevel and Hierarchical Models
Project, Part I
Due: FRIDAY NOVEMBER 18.

Please submit as a single pdf on Blackboard.

1 Exploratory Data Analysis [20 pts]

Number of questions per player. From Figure 1, we can see that the total number of questions attempted by each player is nearly the same as the unique number of questions attempted, except for players who tried all 20 questions; in this case they may have repeated questions for up to about 100 questions attempted. The distribution of questions per player is U-shaped, with the most players seeing 1–5 or 16–20 questions, and fewer seeing intermediate numbers of questions. Similarly, most players see 20 or fewer total questions, with a long tail toward higher numbers of questions; there is an interesting spike at 52 total questions, but there is not enough information in the data or description of the problem to understand why it is there.

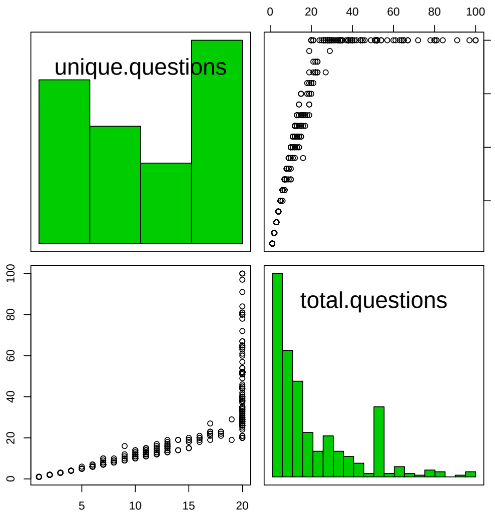


Figure 1: Count of unique questions, and total questions, tried by each player.

"Experiments", players & questions. By experimenting with various tables cross-classifying experimentName with LevelName, isGuidesEnabled, currQuestion and SID, it was possible to determine that

- The various experimentName values ("10thsG", "3rdsG", "NoG", etc.) show whether guides (tick marks) are displayed on the on-screen nunberline, at intervals of 1/10, 1/3, 1/4, 1/2, or not at all. The designation AB and BA indicate something about question order.
- Any item could appear in any experiment. Most players participated in only one experiment; 29 of the 414 players participated in 2–4 experiments, but this appears to be happenstance rather than a designed part of the data collection.

Relation of players to ip numbers. Most players (373) used a single ip address to play the game; 41 players used 2–5 ip numbers. If we reduce to the first two octets of the ip address (variable ip2) we see that 383 players used ip addresses within the same first two octets; 29 used ip addresses with different pairs of initial octets, and 2 used ip addresses with three pairs. Of the 229 unique initial octet pairs, 167 had just a single player, 30 had 2 players, 10 had 3 players, 16 had 4–6 players, and just 6 had 7–39 players. Thus there is no simple nesting, either of players within ip addresses, or ip addresses within players.

Proportion-correct and successful fraction of players. I need a shorter name for "proportion of players who got a question right". I will call it the question's "easiness". Figure 2 shows the distribution of proportion-correct scores for players, and the easiness of each question.

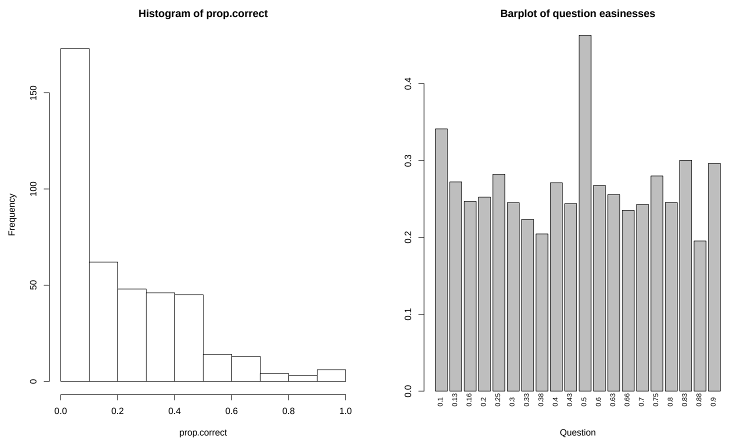


Figure 2: Distribution of players' proportion-correct scores, and of question easiness (proportion of players who got the question right).

We can see that the questions are generally very hard for the players. Further tablation shows that 126 players have a score of 0; the remaining players were distributed fairly evenly across the proportion-correct range, except that 10 or more players had each of the scores 1/7, 1/6, 1/4, 1/3 and 1/2—these are mostly players who played a small multiple k of 7, 6, 4, 3, or 2 questions, and got k right; most often $k = 1$. Correspondingly, most questions had an easiness of 0.3 or less; a somewhat higher proportion of players got certain "landmark" questions right—e.g. 0.10, 0.25, 0.50, 0.75, and 0.90.

Reaction times. From Figure 3 we can see that the average reaction time for most players is 3–4 seconds; a similar graph (not shown) shows that indeed the mean reaction time per question varies little from about 3.7 seconds. (varying little from question to question), but there is a small "spike" around 10 seconds. Exploring the distribution of reaction times for each question (figure not shown because of space limitations), we see that every question has a bump at 10 seconds; this corresponds to the "timeLimit" variable, which sets a limit of 10 seconds on any player's response[1].

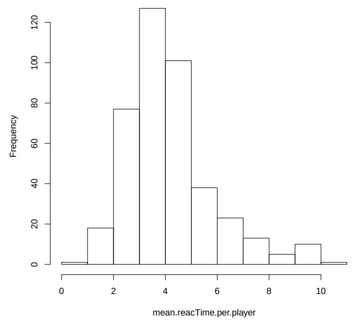


Figure 3: Distribution of players' mean reaction times.

[1]Note that for question 0.88, the time limit of 10 seconds was not imposed for at least one player!

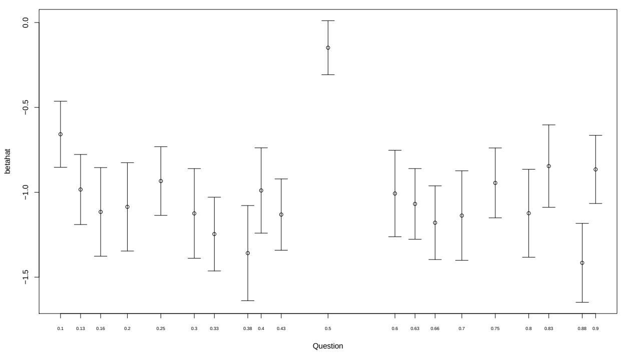
2 Logistic regression for question difficulty [30 pts]

2(a). Figure 4(a) provides an extract of the R summary of the fitted model (where `cq` = `factor(currentQuestion)`).

```
glm(formula = resp ~ cq - 1, family = binomial, data = dec.data)

       Estimate Std. Error z value Pr(>|z|)
cq0.1  -0.65817    0.09740  -6.758 1.40e-11 ***
cq0.13 -0.98373    0.10320  -9.532  < 2e-16 ***
cq0.16 -1.11556    0.13047  -8.550  < 2e-16 ***
cq0.2  -1.08587    0.13012  -8.345  < 2e-16 ***
cq0.25 -0.93378    0.10121  -9.226  < 2e-16 ***
cq0.3  -1.12459    0.13203  -8.518  < 2e-16 ***
cq0.33 -1.24619    0.10869 -11.466  < 2e-16 ***
cq0.38 -1.35857    0.14015  -9.694  < 2e-16 ***
cq0.4  -0.98941    0.12557  -7.879 3.29e-15 ***
cq0.43 -1.13140    0.10498 -10.777  < 2e-16 ***
cq0.5  -0.14830    0.07959  -1.863   0.0624 .
cq0.6  -1.00726    0.12749  -7.901 2.77e-15 ***
cq0.63 -1.06860    0.10409 -10.266  < 2e-16 ***
cq0.66 -1.17935    0.10853 -10.866  < 2e-16 ***
cq0.7  -1.13733    0.13182  -8.628  < 2e-16 ***
cq0.75 -0.94489    0.10296  -9.177  < 2e-16 ***
cq0.8  -1.12361    0.12951  -8.676  < 2e-16 ***
cq0.83 -0.84582    0.12138  -6.968 3.21e-12 ***
cq0.88 -1.41575    0.11622 -12.181  < 2e-16 ***
cq0.9  -0.86537    0.10039  -8.620  < 2e-16 ***
```

(a) Fitted coefficients ($\hat{\beta}$'s).



(b) CI's for coefficients ($\hat{\beta}$'s).

Figure 4: (a) Partial summary of fitted logistic regression for 2(a); (b) Confidence intervals for coefficients.

Fitting the model without the intercept means that each coefficient is exactly the log-odds of getting the corresponding question correct (see also the discussion of the plot for 2(b) below). We see that all of the log-odds are negative, and significantly so except for cq0.5, so the probability of getting any question right is substantially less than 0.5.

With the model set up this way we can also easily extract confidence intervals for the log-odds, of the form "(Estimate) ± 2·(Std. Error)", to make a plot like like Figure 4(b). We see that most of the questions are not significantly different from one another in difficulty, except that (a) locating a submarine at 0.5 on the number line is much easier than any other question; and (b) locating submarines at unfamiliar values like 0.38 and 0.88 is somewhat harder than many other questions.

Residual plots and binned residual plots are not very useful for assessing fit here (see also the discussion of residuals for several models on page 8 below). Conventional residual plots fail because of the 0/1 nature of the `resp` variable. A binned plot (Figure 8 below) does not communicate much because when all the expected values (probabilities of correct response) are less than 0.5, the data has more 0's than 1's so the residuals (data − expected value) are predominantly negative, causing the binned average residual to always be negative.

2(b). The plot of $\hat{\beta}$'s from the model in 2(a) against the logit of easiness (proportion of players who got the question right) provides a perfect linear fit; see Figure 5. The table below shows the values of easiness, $\hat{\beta}$ and logit(easiness) for each question; we see that logit(easiness) is exactly the $\hat{\beta}$ value.

```
  cq  easiness    betahat logit(easiness)
0.1  0.3411514 -0.6581675      -0.6581675
0.13 0.2721519 -0.9837320      -0.9837320
0.16 0.2468354 -1.1155618      -1.1155618
0.2  0.2523962 -1.0858733      -1.0858733
0.25 0.2821577 -0.9337839      -0.9337839
0.3  0.2451613 -1.1245878      -1.1245878
0.33 0.2233607 -1.2461883      -1.2461883
0.38 0.2044728 -1.3585698      -1.3585698
0.4  0.2710280 -0.9894130      -0.9894130
0.43 0.2439024 -1.1314021      -1.1314021
0.5  0.4629921 -0.1483027      -0.1483027
0.6  0.2675159 -1.0072625      -1.0072625
0.63 0.2556701 -1.0685964      -1.0685964
0.66 0.2351695 -1.1793478      -1.1793478
0.7  0.2428115 -1.1373268      -1.1373268
0.75 0.2799145 -0.9448856      -0.9448856
0.8  0.2453416 -1.1236136      -1.1236136
0.83 0.3003096 -0.8458240      -0.8458240
0.88 0.1953291 -1.4157476      -1.4157476
0.9  0.2962185 -0.8653706      -0.8653706
```

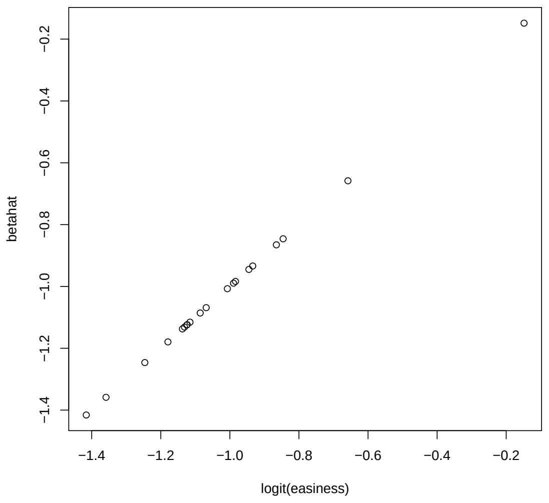


Figure 5: Plot of $\hat{\beta}$'s from the model in 2(a) vs the logit of easienss (proportion of players who got each question right).

The correlation between $\hat{\beta}$ and easiness is quite good (0.996), but the correlation between $\hat{\beta}$ and logit(easiness) is exactly 1.

2(c). I used the `stepAIC` function in `library(MASS)` to search for models with the following variables: `experimentName`, `levelName`, `isGuidesEnabled`, `currentQuestion`, `answerDec`, `avgAccuracy`, `avgTime`, `bestTime`, `currentStarCount`, `sound`, `reacTime`. I omitted variables from dec.data that did not have multiple values (like `timeLimit`), and factors with 100's of levels (like `ip`). I also omitted `currentAccuracy` and `hitType`, since they are perfect classifiers of `resp`. I also omitted `answerDec`, since `answerDec=-1` whenever `hitType="Time Out!!"`.

The final logistic regression model produced by `stepAIC` was

```
    resp ~ cq + avgAccuracy + currentStarCount + avgTime +  levelName + reacTime
```

A binned residual plot (Figure 8 below) suggests much better fit than the model of 2(a), although it is far from perfect (again, see the discussion of residuals for several models on page 8 below), but the difference in AIC is enormous; AIC for the model in 2(a) is 9587, and for this model is it 7363. Parameter estimates and interpretations follow:

- `cq`, the individual effect for each question, is still significant. Again, 0.5 is the easiest question, with the others all more difficult.

- Variables related directly to overall proficiency, `averageAccuracy` and `currentStarCount` (cumulative number of resp=1 over player's session with game), significantly increase the probability of a correct response.

- `levelName` is a significant effect, with all levels more difficult than the baseline level `Decimal_4thsG`. Interestingly, `isGuidesEnabled` is not in the final model, probably because `levelName` is a perfect classifier for it.

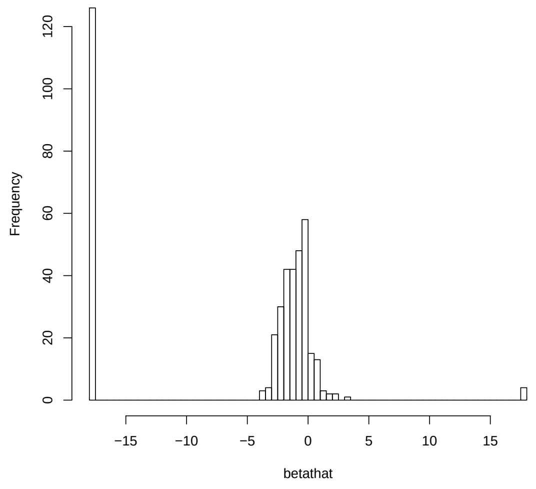
- Interestingly, `avgTime` has a significant positive effect, and `reacTime` has a significant negative effect. It may be that longer reaction times are actually helfpul in producing correct answers (positive effect of `avgTime`), but the negative effect of `reacTime` is due to long reaction times when `hitType="Time Out!!"`.

```
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)              -6.334088   0.292354 -21.666  < 2e-16 ***
cq0.13                   -0.547639   0.174300  -3.142 0.001678 **
cq0.16                   -0.684346   0.194599  -3.517 0.000437 ***
cq0.2                    -0.742743   0.196255  -3.785 0.000154 ***
cq0.25                   -0.481506   0.171948  -2.800 0.005105 **
cq0.3                    -0.862006   0.198259  -4.348 1.37e-05 ***
cq0.33                   -0.713667   0.170928  -4.175 2.98e-05 ***
cq0.38                   -1.222396   0.204483  -5.978 2.26e-09 ***
cq0.4                    -0.585054   0.191788  -3.051 0.002284 **
cq0.43                   -0.738801   0.174627  -4.231 2.33e-05 ***
cq0.5                     0.624137   0.152700   4.087 4.36e-05 ***
cq0.6                    -0.657320   0.195278  -3.366 0.000762 ***
cq0.63                   -0.547422   0.167755  -3.263 0.001102 **
cq0.66                   -0.860831   0.176823  -4.868 1.13e-06 ***
cq0.7                    -0.842317   0.200023  -4.211 2.54e-05 ***
cq0.75                   -0.397344   0.167856  -2.367 0.017925 *
cq0.8                    -0.675758   0.197280  -3.425 0.000614 ***
cq0.83                   -0.480265   0.189801  -2.530 0.011394 *
cq0.88                   -1.013481   0.178419  -5.680 1.34e-08 ***
cq0.9                    -0.295401   0.173489  -1.703 0.088622 .
avgAccuracy               0.061813   0.002763  22.369  < 2e-16 ***
currentStarCount          0.244920   0.012177  20.114  < 2e-16 ***
avgTime                   0.285496   0.030764   9.280  < 2e-16 ***
levelNameDecimal_3rdsG   -0.770360   0.131274  -5.868 4.40e-09 ***
levelNameDecimal_4thsG   -0.631246   0.123603  -5.107 3.27e-07 ***
levelNameDecimal_MidG    -0.712867   0.114132  -6.246 4.21e-10 ***
levelNameDecimal_NoG     -0.825701   0.104304  -7.916 2.45e-15 ***
levelNameDecimal_pretestA -0.472582   0.130918  -3.610 0.000306 ***
levelNameDecimal_pretestB -0.646577   0.135144  -4.784 1.72e-06 ***
reacTime                 -0.112872   0.021679  -5.207 1.92e-07 ***
```

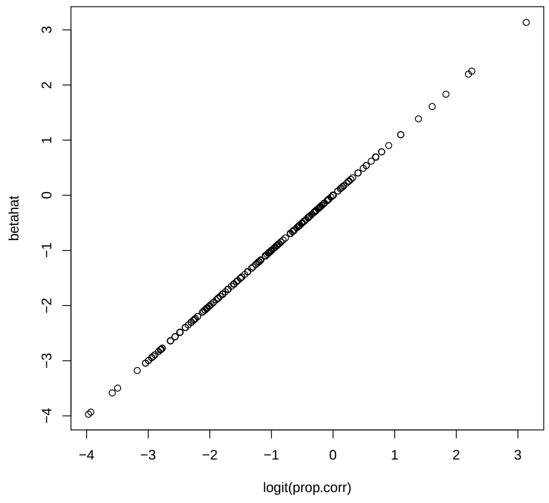
3 Logistic regression for player proficiency [20 pts]

3(a). There are too many coeficients (414!) in the model `glm(resp ~ sID - 1, family=binomial, data=dec.data)`, where `sID = factor(SID)`, to print them all out, but Figure 6(a) presents a histogram. The coefficients greater than 15 or less than −15 correspond to `prop.correct` = 1 or 0, respectively. A binned residual plot here looks a bit like the plot for model 2(a) (see page 8 below); very structured but not suggesting much about fit other than the fact that the number of 0's in the data decreases as the expected value (probability of correct response) increases.

3(b). For the plot of coefficients against proportion correct, I have excluded proportion correct = 0 or 1, since we can't calculate logit(0) or logit(1). The plot of coefficients against proprtion correct shows some curvature, although the correlation is a very high 0.97. The plot of coefficients against logit(proportion correct) is perfectly linear, with correlation 1. Indeed, excluding values for which the proportion correct is 0 or 1, the maximum difference between the coefficient estimate and logit(proportion correct) is $5.77316 \cdot 10^{-15}$, essentially zero except for machine calculation error. Figure 6(b) shows the plot.



(a) Histogram of coefficients from the fitted model in 3(a).



(b) Plot of coefficients vs. logit(proportion correct).

Figure 6: Plots of coefficients from the fitted logistic regression `resp ~ sID - 1`, for section 3.

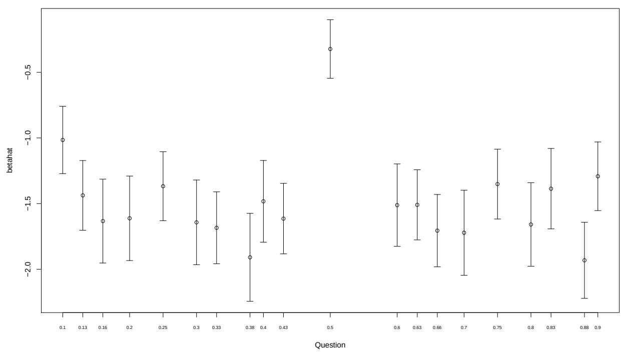
4 Mixed Effects Models [40 pts]

4(a). Figure 7(a) provides a partial summary of the fitted model `glmer(resp ~ cq - 1 + (1|sID), family=binomial, data=dec.data)`:

```
Random effects:
 Groups Name        Variance Std.Dev.
 sID    (Intercept) 1.362    1.167
Number of obs: 8257, groups:  sID, 414

Fixed effects:
       Estimate Std. Error z value Pr(>|z|)
cq0.1   -1.0154     0.1283  -7.917 2.44e-15 ***
cq0.13  -1.4374     0.1327 -10.831  < 2e-16 ***
cq0.16  -1.6329     0.1597 -10.227  < 2e-16 ***
cq0.2   -1.6118     0.1608 -10.022  < 2e-16 ***
cq0.25  -1.3676     0.1312 -10.425  < 2e-16 ***
cq0.3   -1.6426     0.1612 -10.192  < 2e-16 ***
cq0.33  -1.6843     0.1369 -12.301  < 2e-16 ***
cq0.38  -1.9091     0.1674 -11.402  < 2e-16 ***
cq0.4   -1.4825     0.1556  -9.530  < 2e-16 ***
cq0.43  -1.6142     0.1342 -12.031  < 2e-16 ***
cq0.5   -0.3230     0.1115  -2.898  0.00375 **
cq0.6   -1.5116     0.1569  -9.635  < 2e-16 ***
cq0.63  -1.5093     0.1335 -11.304  < 2e-16 ***
cq0.66  -1.7058     0.1377 -12.390  < 2e-16 ***
cq0.7   -1.7218     0.1621 -10.624  < 2e-16 ***
cq0.75  -1.3513     0.1329 -10.165  < 2e-16 ***
cq0.8   -1.6587     0.1590 -10.432  < 2e-16 ***
cq0.83  -1.3862     0.1531  -9.057  < 2e-16 ***
cq0.88  -1.9317     0.1448 -13.342  < 2e-16 ***
cq0.9   -1.2919     0.1306  -9.888  < 2e-16 ***
```

(a) Random effects (η's) and fixed effects ($\hat{\beta}$'s) summary



(b) Confidence intervals for fixed effects ($\hat{\beta}$'s).

Figure 7: (a) Partial summary of fit for mixed effects logistic regression for 4(a); and (b) confidence intervals for fixed effects.

As expected, the fitting algorithm complained that it failed to converge in 10000 evaluations, so we should be careful about how much we trust these parameter estimates. As a quick check to see tha the model fit is not entirely crazy, Figure 7(b) presents a comparison of the $\hat{\beta}$'s for questions, similar to Figure 4(b) above. Figures 4(b) and 7(b) look remarkably similar, which gives us some confidence the the fitted model here is not useless.

Aside. It is interesting to compare the four models we have fitted so far. Figure 8 presents binned residual plots for all four models. The increasing pattern in all four residual plots is due to the fact that when the expected value (probability of a correct answer) is low, there will be many zeros in the data, and the residuals (data − expected value) will be predominantly negative, whereas when the expected value (probability of a correct answer) is high, there will be many one's in the data and the residuals will be predominantly positive. Generally the residuasl for the "final model" of 2(c) and the residuals for the mixed effects model of 4(a) look better.

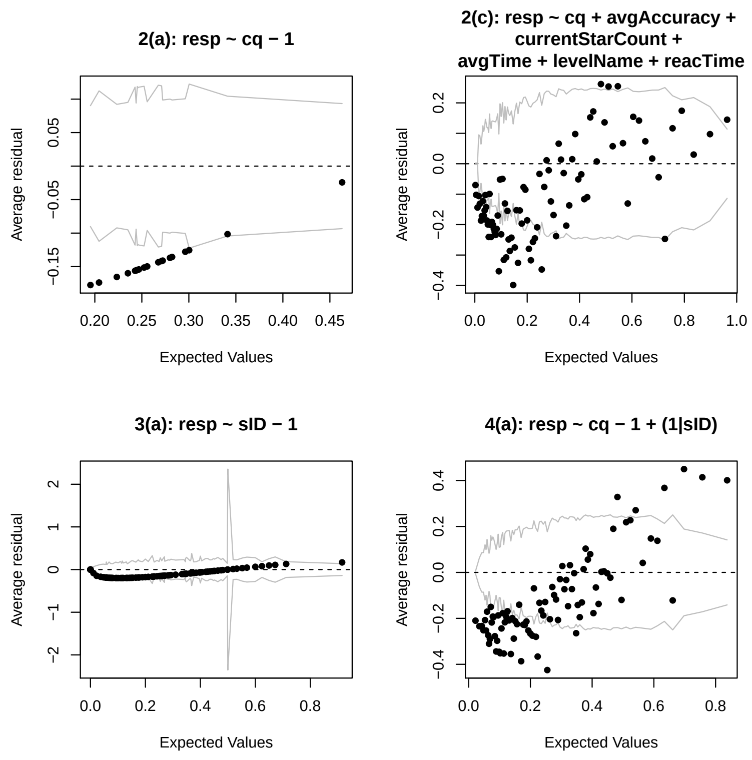


Figure 8: Residual plots for the models from 2(a), 2(c), 3(a) and 4(a), respectively.

.

Comparing AIC and BIC for the four models,

```
    glm.2a  glm.2c   glm.3a glmer.4a
AIC 9586.96 7362.74  8666.51  8613.14
BIC 9727.34 7573.31 11572.30  8760.53
```

we see that the "final model" of 2(c), `resp ~ cq + avgAccuracy + currentStarCount + avgTime + levelName + reacTime`, is preferred by both AIC and BIC; this is consistent with the somewhat better residual plot for this model in Figure 8. The differences in AIC and BIC among the models are dramatic, but it is still interesting that the mixed effects model `resp ~ cq - 1 + (1|sID)` from 4(a) does second-best; it seems like the random effect for player in this model are picking up some of the same variation as the various player covariates (`avgAccuracy, currentStarCount, avgTime, reacTime`) in model 2(c).

4(b) and 4(c). Figure 9 provide plots of fixed effects vs. logit(easiness), and random effects vs. logit(proportion correct), for the model of 4(a). For the proportion correct plots, again we have excluded the players with proportion correct 0 or 1. Although the differences between the plots using logits vs raw proportions are not that great here, logits are still somewhat more linearly related to the $\hat{\beta}$'s and η's than the raw proportions.

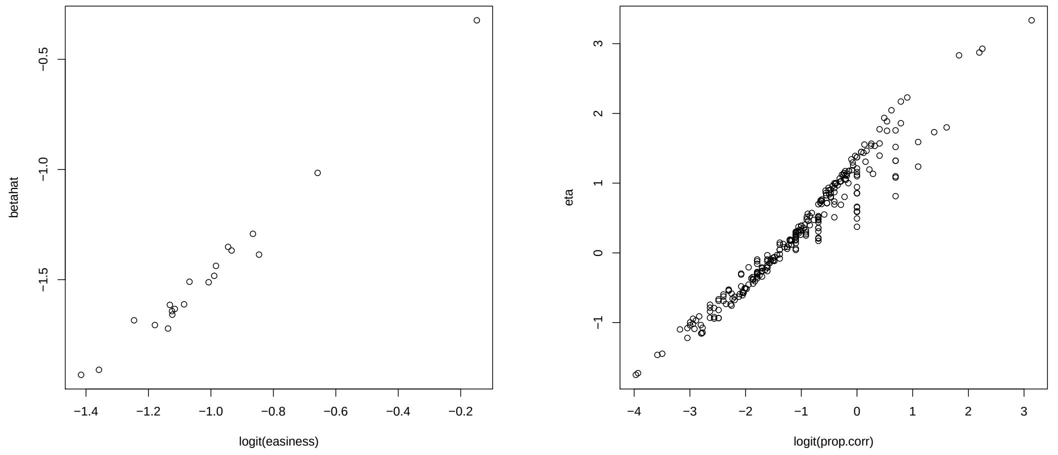


Figure 9: Plots of fixed and random effects vs. logits of sample proportions, for the model of 4(a).

In hierarchical form, the model `glm(resp ~ cq - 1 + (1|sID), family=binomial, data=dec.data)` can be written as

$$\begin{aligned} \text{logit}(p_i) &= \eta_{j[i]} + \beta_{k[i]}\ , \quad i = 1, \ldots, 8257 \\ \eta_j &\overset{iid}{\sim} N(0, \tau^2)\ , \quad j = 1, \ldots, 414 \end{aligned}$$

with fixed effect coefficients β_k, $k = 1, \ldots, 20$ for questions, where $p_i = P[\text{resp}_i = 1]$ on the i^{th} line of the dataset. Thus, logit(P[player j responds correctly to question k]) = $\eta_j + \beta_k$. Thus, for example[2],

$$(\text{easiness of question } k_0) = \frac{1}{n_{k_0}} \sum_{i:k[i]=k_0} p_i = \frac{1}{n_{k_0}} \sum_{i:k[i]=k_0} \text{expit}(\eta_{j[i]} + \beta_{k_0})\ ,$$

where n_k is the number of players who answers question k, so that

$$\text{logit}(\text{easiness of question } k_0) = \text{logit}\left[\frac{1}{n_{k_0}} \sum_{i:k[i]=k_0} \text{expit}(\eta_{j[i]} + \beta_{k_0})\right]\ .$$

If the η's were all equal to zero, the right hand side would just be (β_{k_0}), and the linearity in the plot of $\hat{\beta}$ vs logit(easiness) would be perfect. When the η's are nonzero (which they are, since η is a random effect with $\tau^2 = 1.362$), it is kind of amazing that the plot is still almost linear. A similar analysis makes the linearity in the plot of η vs logit(proportion correct) kind of amazing. . .

[2]Recall that expit(x) and logit(p) are inverse functions.

4(d) and 4(e). Since the model fits are slow it's important to try to limit the number of different models fitted, but still get a good sense of variable selection. I started by adding all of the variables from the final model in 2(c) to the glmer model fitted in 4(a). The idea was to see if any of those variables are no longer sigificant predictors when the `(1|sID)` random effect is also in the model. Here is the result of the fit:

```
> glmer.2a <- glmer(resp ˜ cq  + avgAccuracy + currentStarCount +
+                       avgTime + levelName + reacTime + (1|sID),
                  family=binomial,data=dec.data)
> summary(glmer.2a)
Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: binomial  ( logit )
Formula: resp ˜ cq + avgAccuracy + currentStarCount + avgTime + levelName +
    reacTime + (1 | sID)
   Data: dec.data

     AIC      BIC   logLik deviance df.resid
  7364.7   7582.3  -3651.4   7302.7     8226

Scaled residuals:
    Min      1Q  Median      3Q     Max
-4.4611 -0.5267 -0.2842  0.4335  8.4255

Random effects:
 Groups Name        Variance Std.Dev.
 sID    (Intercept) 0        0
Number of obs: 8257, groups:  sID, 414

Fixed effects:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)               -6.334088   0.292355 -21.666  < 2e-16 ***
cq0.13                    -0.547639   0.174302  -3.142 0.001679 **
cq0.16                    -0.684346   0.194603  -3.517 0.000437 ***
cq0.2                     -0.742743   0.196259  -3.784 0.000154 ***
cq0.25                    -0.481506   0.171949  -2.800 0.005106 **
cq0.3                     -0.862006   0.198262  -4.348 1.38e-05 ***
cq0.33                    -0.713667   0.170930  -4.175 2.98e-05 ***
cq0.38                    -1.222396   0.204483  -5.978 2.26e-09 ***
cq0.4                     -0.585054   0.191788  -3.051 0.002284 **
cq0.43                    -0.738801   0.174630  -4.231 2.33e-05 ***
cq0.5                      0.624137   0.152702   4.087 4.36e-05 ***
cq0.6                     -0.657320   0.195280  -3.366 0.000763 ***
cq0.63                    -0.547422   0.167758  -3.263 0.001102 **
cq0.66                    -0.860831   0.176826  -4.868 1.13e-06 ***
cq0.7                     -0.842317   0.200024  -4.211 2.54e-05 ***
cq0.75                    -0.397344   0.167859  -2.367 0.017927 *
cq0.8                     -0.675758   0.197281  -3.425 0.000614 ***
cq0.83                    -0.480265   0.189803  -2.530 0.011395 *
cq0.88                    -1.013481   0.178421  -5.680 1.34e-08 ***
cq0.9                     -0.295401   0.173490  -1.703 0.088624 .
avgAccuracy                0.061813   0.002763  22.370  < 2e-16 ***
currentStarCount           0.244920   0.012177  20.114  < 2e-16 ***
avgTime                    0.285496   0.030765   9.280  < 2e-16 ***
levelNameDecimal_3rdsG    -0.770360   0.131276  -5.868 4.40e-09 ***
levelNameDecimal_4thsG    -0.631246   0.123605  -5.107 3.27e-07 ***
levelNameDecimal_MidG     -0.712867   0.114134  -6.246 4.21e-10 ***
levelNameDecimal_NoG      -0.825701   0.104306  -7.916 2.45e-15 ***
levelNameDecimal_pretestA -0.472582   0.130919  -3.610 0.000307 ***
levelNameDecimal_pretestB -0.646577   0.135145  -4.784 1.72e-06 ***
reacTime                  -0.112872   0.021679  -5.206 1.93e-07 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Correlation matrix not shown by default, as p = 30 > 12.
Use print(x, correlation=TRUE)  or
         vcov(x)        if you need it

convergence code: 0
Model is nearly unidentifiable: very large eigenvalue
 - Rescale variables?
Model is nearly unidentifiable: large eigenvalue ratio
 - Rescale variables?
```

We immediately see two things: First, all of the variables that were in the final model in 2(c) are still significant predictors in this model; second, the random effect has essentially zero variance (up to round-off error for printing, anyway). That is, when all of the predictors from 2(c) are in the model, the random effect isn't needed anymore; the model `glmer.2` is essentially the same as the `final.model` of 2(c). (The complaint at the end about an unidentifiable model is due to the vanishing random effect; see for example `https://stat.ethz.ch/pipermail/r-sig-mixed-models/2014q1/021736.html`).

I did go ahead and try to fit random effects for the full or partial IP numbers, both with and without the sID random effect.

```
> glmer.3 <- update(glmer.2a, . ˜ . + (1|ip3))
> summary(glmer.3)$varcor
 Groups Name        Std.Dev.
 sID    (Intercept) 0.00043513
 ip3    (Intercept) 0.07757963
> glmer.4 <- update(glmer.2a, . ˜ . + (1|ip))
> summary(glmer.4)$varcor
 Groups Name        Std.Dev.
 sID    (Intercept) 0.00022553
 ip     (Intercept) 0.04846029
> glmer.3a <- update(glmer.2a, . ˜ . + (1|ip3) - (1|sID))
> summary(glmer.3a)$varcor
```

```
 Groups Name        Std.Dev.
 ip3    (Intercept) 0.077649
> glmer.4a <- update(glmer.2a, . ~ . + (1|ip) - (1|sID))
> summary(glmer.4a)$varcor
 Groups Name        Std.Dev.
 ip     (Intercept) 0.048578
```

In each case, the variance of the random effects is still quite small (all model fits complained of the same unidentifiability problem, for example). A quick check of AIC and BIC confirm that none of these models does any better than `glmer.2a`, which is the same as the `final.model` from 2(c).

```
    glmer.2a  glmer.3  glmer.4 glmer.3a glmer.4a
AIC 7364.743 7366.292 7366.715 7364.292 7364.715
BIC 7582.327 7590.895 7591.317 7581.876 7582.298
```

Thus, the overall conclusion is that random effects for players and/or ip numbers are not needed, if we include the covariates from `final.model` in 2(c).

5 Summary [10 pts]

The data from Dr. Lomas consist of responses from 414 players on up to 20 unique questions in the game *Battleship Numberline*, in which players try to locate a hidden submarine on a unit-interval numberline by clicking on the numberline at the location given by a decimal fraction. Not all players saw all 20 questions, and some players saw questions repeatedly, so that players saw as few as one or as many as 100 questions (Figure 1 above); in all, there were 8257 instances of players answering questions.

Players' accuracy in locating the submarine was recorded as a four-level "partial credit" categorical variable (Perfect Hit!!, Partial Hit!!, Near Miss!!, Miss!!), plus a fifth category for time-out (Time Out!!); however, for this analysis we considered only the binary outcome resp=1 if a Perfect Hit!! was recorded, and resp=0 otherwise.

Players played the game from all over the world, as indicated by the 280 unique IP addresses of internet connections that they played the game on. Most players (373) used a single ip address; 41 players used 2–5 ip addresses.

In addition to unique identifiers for questions (currentQuestion) and player (SID), the following variables were thought to be interesting:

currentQuestion Decimal location of hidden submarine on numberline.
isGuidesEnabled Indicates whether tick marks are drawn on the numberline.
experimentName The various experimentName values("10thsG", "3rdsG", "NoG", etc.) show whether guides (tick marks) are displayed on the on-screen nunberline, at intervals of 1/10, 1/3, 1/4, 1/2, or not at all. The designation AB and BA indicate something about question order.
levelName Groupings of experimentName.
avgAccuracy Running average of distance on numberline between player's location and true location of the submarine, during player's session.
bestTime Running best (shortest) reaction time, during player's session.
currentStarCount Running number of correct responses, during player's session.
sound Indicates whether sound (clicks, positive and negative feedback on player's response, etc.) is turned on.
reacTime Number of seconds player took to answer question.

We fitted several logistic regression and mixed effects logistic regression models. The two most successful models for the probability p_i that $\text{resp}_i = 1$ ($i = 1, \ldots, 8257$), were

1. `resp ~ cq - 1 + (1 | sID)`
   This model asserts that $p_i = P[\text{resp}_i = 1]$ can be modelled in terms of effects for the player and question that generates resp_i:

$$\log p_i/(1 - p_i) = \eta_{\text{player}[i]} + \beta_{\text{question}[i]} \ .$$

   Here, $\beta_{question}$ is a fixed effect parameter for each of the 20 unique questions, and η_{player} is a random effect which models the dependence between responses from the same player (if we know that a player did well on the first three questions in a session, we can guess that the player will continue to do well, for example).

The fixed effect parameters $\beta_{question}$ tell us how difficult or easy each question is. For example, Figure 7(b) shows confidence intervals for the log odds of getting each question right, as a function of the location of the sub on the number line. We see that questions at "landmark" values such as 0.1, 0.9, and especially 0.5, are much easier than questions at other values; and unusual values like 0.88 are especially hard. Moreover, since the log-odds are all negative, the questions are substantially difficult for the player; a player of middling proficiency ($\eta = 0$) has less than 1/2 probability of getting each question right.

2. `resp ~ cq + avgAccuracy + currentStarCount + avgTime + levelName + reacTime`
   This model was obtained by stepwise regression-style variable selection, and asserts that $p_i = P[\text{resp}_i = 1]$ can be modeled as

$$\begin{aligned}\log p_i/(1-p_i) &= \beta_{0,question[i]} + \beta_1(\text{avgAccuracy})_i + \beta_2(\text{currentStarCount})_i \\ &\quad + \beta_3(\text{avgTime})_i + \beta_{4,levelName[i]} + \beta_5(\text{reacTime})_i \ .\end{aligned}$$

   As before, $\beta_{0,question}$ is a fixed effect for each unique question, but now the random effect has been replaced with a fixed effect $\beta_{4,levelName}$ for guide conditions (presence and location of tickmarks on numberline), and several continuous predictors related to players' performance: avgAccuracy, currentStarCount, avgTime and reacTime. The parameter estimates are provided on page 5 above.

   As performance improves (higher avgAccuracy and currentStarCount), the probability of the player providing a correct response increases. avgTime and reacTime provide somewhat contradictory evidence: longer avgTime, but shorter reacTime, is associated with a higher probability of correct response. What may be going on is that long reacTime's are associated with the "Time Out!!" coding, which is an incorrect response in our analysis; reacTime may be more sensitive to this than its average avgTime.

Model 2 provides substantially better fit (see the analysis on page 8) and also has good interpretive power (the predictors mean something!) and so it is probably preferable for many purposes. Model 1 might be preferable if we were trying to rank players by proficiency; the random effect η_{player} provides a common scale on which to rank players, regardless of how many or which questions they saw.

Finally we explored whether question easiness (proportion of players who get the question right) is related to other variables in the data set. Most of these variables seem to be more associated with players or experimental conditions than with questions. Two variables that arguably could be related to the questions themselves are: (a) location of the submarine on the number line; and (b) the mean reaction time of all players encountering that question. Question easiness has very little relationship with the location of the sub on the numberline, but it does have a modest relationship with mean reaction time (correlation ≈ 0.40).

Appendix I: List of references and materials used.

CRAN (2016). The Comprehensive R Archive Network[3] (https://cran.r-project.org/). Author.

Gelman, A. & Hill, J. (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models.* NY: Cambridge Univ Press.

Lomas, D., Patel, K., Forlizzi, J.L., and Koedinger, K.R. (2013). Optimizing Challenge in an Educational Game Using Large-Scale Design Experiments. Paper presented at CHI 2013, Paris, France. Obtained online at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.480.2493&rep=rep1&type=pdf

Lynch, Scott M. (2007). *Introduction to Applied Bayesian Statistics and Estimation for Social Scientists.* New York: Springer.

[3]For documentation on various packages.

Appendix II: Additional material, including R code.

Here is all the code that I used to prepare these solutions. I have indicated in comments which part of the project each block of code was for… (This is much more than I would expect you to include, and more than I would grade, but I thought you might find it interesting.

```
library(arm)      # generally useful tools for lm, glm and (g)lmer models.

library(car)      # for scatterplot matrix with marginal histograms
library(plotrix)  # for plotting confidence intervals

dec.data <- read.csv("dec-data.csv",header=T)

cq <- factor(dec.data$currentQuestion)
sID <- factor(dec.data$SID)

str(dec.data)

# 'data.frame':   8257 obs. of  32 variables:
#
#  $ X                    : int  1 2 3 4 5 6 7 8 9 10 ...
#                           unique row number for every row in the data set
#
#* $ SID                  : int  161461 161461 ...
#                           unique identifier for each player; see also sID
#
#* $ ip                   : Factor w/ 280 levels "108.66.1.69 ",...
#                           unique identifier for each computer used;
#                           see also ip2 and ip3
#
#  $ userName             : Factor w/ 414 levels "User 161461",...
#                           same as SID
#
#* $ experimentName       : Factor w/ 15 levels "Decimal_10thsG",...
#                           name of experiment for this participant x question
#                           interaction
#
#* $ levelName            : Factor w/ 7 levels "Decimal_10thsG",...
#                           groups of experiment names
#
#* $ currentLevelNo       : int  2 2 2 2 3 3 3 3 3 3 ...
#                           game level reported to player
```

```
#
#  $ totalTrials          : int  20 20 20 20 20 20 20 20 20 20 ...
#                           maximum number of questions allowed per student
#
#* $ isGuidesEnabled      : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
#                           are hashmarks printed on the numberline
#                           that players see?
#
#* $ currentQuestion      : num  0.83 0.63 0.43 0.5 ...
#                           correct location of the submarine as a number on
#                           the number line. These uniquely identify the
#                           questions that students see.
#
#  $ answerByUser         : Factor w/ 7372 levels "(-1#-1)","(0 # 0)",...
#                           internal coding for location of player's mouse
#                           click on screen
#
#* $ answerDec            : num  0.849 0.591 0.402 0.497 0.58 ...
#                           location of player's mouse click as a number on
#                           the number line
#
#* $ hitType              : Factor w/ 5 levels "Miss!!","Near Miss!!",...
#                           "Perfect Hit!!" = answerDec is close enough to
#                           currentQuestion to count as a "correct answer";
#                           all other responses are "wrong" or "partial credit"
#
#  $ currentTrialStartTime: Factor w/ 7145 levels "Fri Nov 11..."...
#                           timestamp at start of current question
#
#  $ currentTrialEndTime  : Factor w/ 6842 levels "-1","Fri Nov 11..."...
#                           timestamp at end of current question
#
#  $ currentAccuracy      : num  98.1 96.1 97.2 99.7 98 ...
#                           a measure of accuracy of player's response
#
#  $ avgAccuracy          : num  97.4 97.3 97.3 97.4 98 ...
#                           running average of "currentAccuracy" over the
#                           questions each player attempted
#
#* $ curReactionTime      : Factor w/ 4046 levels "0.016 secs","0.024 secs",...
#                           The time it took for player to produce an answer
#                           on this question; see also "reacTime" below.
```

```
#
#  $ totalTime            : Factor w/ 8147 levels "10.324 Secs",...
#                           Not sure; this may be the time from the start
#                           of one question to start of the next question...
#
#  $ timeLimit            : int  10 10 10 10 10 10 10 10 10 10 ...
#                           maximum number of second allowed, per question.
#                           if curReactionTime > timeLimit, then hitType =
#                           "Time Out!!", although this rule does not seem
#                           to be strictly enforced.
#
#  $ avgTime              : num  2.49 2.47 2.47 2.44 3.54 ...
#                           Running average of curReactionTime through each
#                           player’s session with the game
#
#  $ bestTime             : num  1.39 1.39 1.39 1.39 3.54 ...
#                           Minimum of curReactionTime through each
#                           player’s session with the game
#
#  $ currentStarCount     : int  13 13 14 15 1 2 2 3 4 5 ...
#                           Cumulative number correct ("Perfect Hit!!") in the
#                           current session
#
#  $ totalStarCount       : int  688 701 715 730 746 748 750 753 757 762 ...
#                           seems to be cumulative sum of currentStarCount...
#
#  $ fireType             : Factor w/ 2 levels "","CLICK": 2 2 2  ...
#                           Did player respond with mouse click or a key press?
#
#  $ sound                : Factor w/ 2 levels "OFF","ON": 2 2 22 2 2 2 2 ...
#                           Is the sound on or off in the game?
#
#* $ resp                 : int  1 0 1 1 1 1 0 1 1 1 ...
#                           resp = 1 if hitType="Perfect Hit!!", else resp = 0.
#
#* $ reacTime             : num  2.05 2.23 2.32 1.96 3.54 ...
#                           numerical value of currentReactionTime
#
#* $ ip2                  : num  12.2 12.2 12.2 12.2 12.2 ...
#                           first two numbers in ip address
#
#* $ ip3                  : Factor w/ 256 levels "108.66.1", ...
```

```
#                           first three numbers in ip address
#
##################################################################

### PART I: EDA
### (two pages)

### How many questions does each player try?

View(dec.data)

# total number of questions each player saw
total.questions <- with(dec.data,sapply(split(SID,SID),length))
# with(dec.data,table(SID)) also works but produces something
# other than a simple vector....

# number of *unique* questions each player saw
unique.questions <- with(dec.data,sapply(split(currentQuestion,SID),
                                         function(x) length(unique(x))))

# from library(car):
scatterplotMatrix(data.frame(unique.questions,total.questions),
                  diagonal="hist",smoother=F,reg.line=F)

cumsum(table(total.questions))

### How are the "experiments" related to questions or players?

# make the R window big to see all of this...
with(dec.data,table(experimentName,levelName))

with(dec.data,table(experimentName,currentQuestion))

player.questions.within.experiment <-
    with(dec.data,split(data.frame(SID,currentQuestion,experimentName),
                        experimentName))

View(player.questions.within.experiment[[1]])

with(dec.data,table(experimentName,isGuidesEnabled))

experiments.per.player <- with(dec.data,split(as.character(experimentName),SID))
experiments.per.player <- lapply(experiments.per.player,
                                 function(x) sort(unique(x)))
experiments.per.player[[1]]
head(experiments.per.player)

num.per.player <- sapply(experiments.per.player,length)
summary(num.per.player)
num.per.player[num.per.player>1]
length(num.per.player[num.per.player>1])
length(num.per.player)
experiments.per.player[num.per.player>1]


### HOw are the players related to ip numbers or partial ip numbers?

ip.per.player <-  with(dec.data,split(as.character(ip),SID))
ip.per.player <-  lapply(ip.per.player,
                                 function(x) sort(unique(x)))
summary(ip.num.per.player <- sapply(ip.per.player,length))
table(ip.num.per.player)

ip3.per.player <-  with(dec.data,split(as.character(ip3),SID))
ip3.per.player <-  lapply(ip3.per.player,
                                 function(x) sort(unique(x)))
summary(ip3.num.per.player <- sapply(ip3.per.player,length))
table(ip3.num.per.player)
ip3.per.player[ip3.num.per.player==2]
ip3.per.player[ip3.num.per.player==3]

ip2.per.player <-  with(dec.data,split(as.character(ip2),SID))
ip2.per.player <-  lapply(ip2.per.player,
                                 function(x) sort(unique(x)))
summary(ip2.num.per.player <- sapply(ip2.per.player,length))
table(ip2.num.per.player)
ip2.per.player[ip2.num.per.player==2]
ip2.per.player[ip2.num.per.player==3]

players.per.ip2 <- with(dec.data,split(SID,as.character(ip2)))
players.per.ip2 <- lapply(players.per.ip2,
                                 function(x) sort(unique(x)))
players.num.per.ip2 <- sapply(players.per.ip2,length)
summary(players.num.per.ip2)
```

```
table(players.num.per.ip2)

length(unique(dec.data$ip2))

### What is the distribution of proportion-correct scores for players?

resp.per.player <- with(dec.data,split(resp,SID))
prop.correct <- sapply(resp.per.player,mean)

### What fraction of players got each question right?

resp.per.question <- with(dec.data,split(resp,currentQuestion))
easiness <- sapply(resp.per.question, mean)

par(mfrow=c(1,2))

hist(prop.correct)
barplot(easiness,names.arg="")
oldmar <- par()$mar
par(mar=c(3,4,4,2)+.1)
text(.6 + (0:19)*1.2,-.02,labels=names(easiness),srt=90,cex=0.75,adj=c(0,0.5))
par(mar=oldmar)
title(main="Barplot of question easinesses",xlab="Question")

(bozo <- table(round(prop.correct,3)))
bozo[bozo>9]

total.questions[prop.correct*7==1]
total.questions[prop.correct*6==1]
total.questions[prop.correct*4==1]
total.questions[prop.correct*3==1]
total.questions[prop.correct*2==1]

### What is the distribution of reaction times across players?
### Across questions?

reacTime.per.player <- with(dec.data,split(reacTime,SID))
mean.reacTime.per.player <- sapply(reacTime.per.player,mean)
sd.reacTime.per.player <- sapply(reacTime.per.player,sd)

reacTime.per.question <- with(dec.data,split(reacTime,currentQuestion))
mean.reacTime.per.question <- sapply(reacTime.per.question,mean)
```

```
sd.reacTime.per.question <- sapply(reacTime.per.question,sd)

par(mfrow=c(1,1))
hist(mean.reacTime.per.player,main="")
# hist(mean.reacTime.per.question,main="")

par(mfrow=c(1,2))

barplot(mean.reacTime.per.question,names.arg="")
oldmar <- par()$mar
par(mar=c(3,4,4,2)+.1)
text(.6 + (0:19)*1.2,-.17,labels=names(easiness),srt=90,cex=0.75,adj=c(0,0.5))
par(mar=oldmar)
title(main="",xlab="Question",ylab="Mean Reaction Time")

barplot(sd.reacTime.per.question,names.arg="")
oldmar <- par()$mar
par(mar=c(3,4,4,2)+.1)
text(.6 + (0:19)*1.2,-.17,labels=names(easiness),srt=90,cex=0.75,adj=c(0,0.5))
par(mar=oldmar)
title(main="",xlab="Question",ylab="SD of Reaction Time")

par(mfrow=c(4,5))
for (i in 1:20) {
    plot(density(reacTime.per.question[[i]]),
         main=names(reacTime.per.question)[i],xlab="",ylab="")
}

###################################################################

### PART II: Logistic regression for question difficulty
### (one page per subpart)

### (a) P(resp=1) ~ currentQuestion - 1; summarize fit & comment on intercept

summary(glm.2a <- glm(resp ~ cq -1, family=binomial, data=dec.data))

# the following sets of residual plots are not particularly helpful in
# assessing the fit:
par(mfrow=c(2,2))
plot(glm.2a)
par(mfrow=c(1,1))
```

```
binnedplot(fitted(glm.2a),resid(glm.2a))

# here's why:
sum(dec.data$resp==0)
# [1] 5990
sum(resid(glm.2a)<0)
# [1] 5990
sum(dec.data$resp==1)
# [1] 2267
sum(resid(glm.2a)>0)
# [1] 2267
#
# Thus, negative residuals will predominate; you can check that the
# same thing happens within bins of the binned plot, so that the
# average residual within each bin is negative.

# here's a plot comparing the coefficients from glm.2a:
est.2a <- coef(summary(glm.2a))
betahat <- est.2a[,1]
LL <- betahat - 2*est.2a[,2]
UL <- betahat + 2*est.2a[,2]
Question <- sort(unique(dec.data$currentQuestion))

# from library(plotrix)
plotCI(Question, betahat, li=LL,ui=UL,axes=F)
box()
axis(2)
axis(1,at=Question,labels=Question,tick=T,cex.axis=0.6)


### (b) plot coeffs vs frac of participants that got question right (or logit
###     of it). Provide better plot; explain why; interpret.

plot(betahat ~ easiness)
plot(betahat ~ logit(easiness))

cbind(easiness,betahat,"logit(easiness)"=logit(easiness))

cor(betahat,easiness)
cor(betahat,logit(easiness))

### (c) variable selection to augment this model (i) parag describing what
###     you did; (ii) parag interpreting fit

names(dec.data)

# the following excludes factors with many levels, like "ip", "SID",
# etc. (these are all player covariates); also excludes
# "currentAccuracy" and "hitType" because they are perfect classifiers
# of "resp" (fitted probabilities near 0 and 1)... you can see this with
#
# with(dec.data,sapply(split(currentAccuracy,resp),range))
# with(dec.data,lapply(split(hitType,resp),table))
#
# "answerDec" was also omitted because answerDec = -1 iff hitType =
# "Time Out!!".  You can see this with
#
# lapply(with(dec.data,split(answerDec,hitType)),summary)

vars.for.varsel <- scan(what="")
resp
experimentName
levelName
isGuidesEnabled
currentQuestion
avgAccuracy
avgTime
bestTime
currentStarCount
sound
reacTime

varsel.data <- cbind(dec.data[,vars.for.varsel],cq=cq)

base.model <- glm(resp ~ cq,family=binomial,data=varsel.data)

upper.formula <- as.formula(paste("~ ",paste(names(varsel.data)[-1],
                                             collapse="+")))

final.model <- stepAIC(base.model,scope=list(lower= ~1, upper = upper.formula),
                       direction="both")

summary(final.model)
```

```
binnedplot(fitted(final.model),resid(final.model))
par(mfrow=c(2,2))
plot(final.model)

rbind("AIC"=cbind("glm.2a"=AIC(glm.2a),"final.model"=AIC(final.model)))

####################################################################

### PART III: logistic regression for player proficiency
### (one page per subpart)

### (a) P(resp=1) ~ SID - 1; summarize fit

summary(glm.3a <- glm(resp ~ sID - 1, family=binomial, data=dec.data))

hist(coef(glm.3a),nclass=100,main="",xlab="betathat")

group <- ifelse(coef(glm.3a)< -5,1,ifelse(coef(glm.3a)>5,3,2))
split(prop.correct,group)

binnedplot(resid(glm.3a),fitted(glm.3a))
par(mfrow=c(2,2))
plot(glm.3a)


### (b) Plot coeff's against proportion correct or logit(prop corr);
###     choose better plot; explain why; interpret

coef.3a <- coef(glm.3a)[group==2]
prop.corr <- prop.correct[group==2]

plot(prop.corr,coef.3a,ylab="betahat")

plot(logit(prop.corr),coef.3a,ylab="betahat")

cor(coef.3a,prop.corr)
cor(coef.3a,logit(prop.corr))

max(abs(coef(glm.3a)-logit(prop.correct)))

max(abs(coef.3a-logit(prop.corr)))
```

```
####################################################################

### PART IV: mixed effects
### (one page per subpart)

### (a) P(correct) ~ currentQuestion + (1|SID)

summary(glmer.1 <- glmer(resp ~ cq - 1 + (1|sID), family=binomial,
                         data=dec.data))

est.3a <- coef(summary(glmer.1))
betahat <- est.3a[,1]
LL <- betahat - 2*est.3a[,2]
UL <- betahat + 2*est.3a[,2]
Question <- sort(unique(dec.data$currentQuestion))

# from library(plotrix)
plotCI(Question, betahat, li=LL,ui=UL,axes=F)
box()
axis(2)
axis(1,at=Question,labels=Question,tick=T,cex.axis=0.6)

# Comparing various residual plots...


par(mfrow=c(2,2))
binnedplot(fitted(glm.2a),resid(glm.2a), main="2(a): resp ~ cq - 1")
binnedplot(fitted(final.model),resid(final.model),
   main="2(c): resp ~ cq + avgAccuracy + \ncurrentStarCount + \navgTime + levelName + reacTime
 )
binnedplot(fitted(glm.3a),resid(glm.3a), main="3(a): resp ~ sID - 1")
binnedplot(fitted(glmer.1),resid(glmer.1), main="4(a): resp ~ cq - 1 + (1|sID)")

# and look at AIC's fun ....

BIC <- function(x) AIC(x,k=log(length(resid(x))))
rbind("AIC"=c(glm.2a=AIC(glm.2a),glm.2c=AIC(final.model),
              glm.3a=AIC(glm.3a),glmer.4a=AIC(glmer.1)),
      "BIC"=c(glm.2a=BIC(glm.2a),glm.2c=BIC(final.model),
              glm.3a=BIC(glm.3a),glmer.4a=BIC(glmer.1)))
```

```
### (b) plot the fixed effects against frac of players who get it right
###     (or logit of same); provide, defend and interpret the better plot

plot(easiness,fixef(glmer.1), ylab="betahat")
plot(logit(easiness),fixef(glmer.1), ylab="betahat")

cor(easiness,fixef(glmer.1))
cor(logit(easiness),fixef(glmer.1))

### (c) plot the random effects against prop correct (or logit of same);
###     provide, defend and interpret the better plot

plot(prop.correct,ranef(glmer.1)$sID[,1], ylab="eta")
plot(logit(prop.correct),ranef(glmer.1)$sID[,1], ylab="eta")

cor(prop.correct,ranef(glmer.1)$sID[,1])
cor(logit(prop.correct),ranef(glmer.1)$sID[,1])

prop.corr <- prop.correct[group==2]
eta <- ranef(glmer.1)$sID[,1][group==2]

plot(prop.corr,eta, ylab="eta")
plot(logit(prop.corr),eta, ylab="eta")

cor(prop.corr,eta)
cor(logit(prop.corr),eta)


### (d) var selection to augment model; describe the approach used and display
###     best model

fits <- list()
for (var in vars.for.varsel[-1]) {
    fits <- c(fits,update(glmer.1, as.formula(paste(".~.+",var))))
    print(var)
}

lapply(fits,summary)

# the following variables are significant predictors, added one at a
# time:
```

```
#
# experimentName, levelName, isGuidesEnabled (yes better), avgAccuracy
# (higher better), avgTime (faster better), bestTime (faster better),
# currentStarCount (more better), sound (off better), reacTime (faster
# better)
#
# only currentQuestion didn't work (collinearity issue).
#

# might be simpler to fit all the predictors from 2(c) and see which
# ones still matter... this is a kind of backwards selection
# process...

glmer.2 <- update(glmer.1, . ~ . + avgAccuracy + currentStarCount +
                               avgTime + levelName + reacTime)

summary(glmer.2)
# this model is more difficult to compare "by eyeball" with
# final.model since I didn't take the intercept out of final.model in
# 2(c)...

glmer.2a <- glmer(resp ~ cq   + avgAccuracy + currentStarCount +
                      avgTime + levelName + reacTime + (1|sID),
                  family=binomial,data=dec.data)

summary(glmer.2a)
# this model is set up like final.model in 2(c) for easier eyeball
# comparisons...

# all the predictors matter, and now the variance component is
# essentially zero!  So we are back to the final model of 2(c).
# Indeed, even the coefficients for the fixed effects are the same as
# they were for final.model...



### (e) Does adding a 2nd intercept grouped by one of the ip variables help?

# from the exploratory analysis we know that ip2 and ip3 give the same
# groups, so we will only try one of them...

glmer.3 <- update(glmer.2, . ~ . + (1|ip3))
```

```
summary(glmer.3)$varcor

glmer.4 <- update(glmer.2, . ~ . + (1|ip))
summary(glmer.4)$varcor

glmer.3a <- update(glmer.2a, . ~ . + (1|ip3) - (1|sID))
summary(glmer.3a)$varcor

glmer.4a <- update(glmer.2a, . ~ . + (1|ip) - (1|sID))
summary(glmer.4a)$varcor

rbind(AIC=c(glmer.2a=AIC(glmer.2a),glmer.3=AIC(glmer.3),
            glmer.4=AIC(glmer.4),glmer.3a=AIC(glmer.3a),
            glmer.4a=AIC(glmer.4a)),
      BIC=c(glmer.2a=BIC(glmer.2a),glmer.3=BIC(glmer.3),
            glmer.4=BIC(glmer.4),glmer.3a=BIC(glmer.3a),
            glmer.4a=BIC(glmer.4a)))

######################################################################

### PART V: Summary (2 pp)

### interesting relationships between question difficulty and other
### variables?

### briefly describe data set, methods, conclusions.

# the following was fun but not very interesting, since most of the
# variables arguably can't affect question easiness except in an
# associational way (maybe easy questions get asked more often when
# there is a pretest, for example)...
long.ease <- currentQuestion
for (i in 1:length(currentQuestion)) {
    long.ease[i] <- easiness[which(names(easiness)==currentQuestion[i])]
}

summary(lm.0 <- lm(logit(long.ease) ~ experimentName + levelName +
isGuidesEnabled + currentQuestion + avgAccuracy + avgTime + bestTime +
currentStarCount + sound + reacTime,data=dec.data))

lm.1 <- stepAIC(lm.0)
```

```
summary(lm.1)

############

# really the only variables that make much sense to be associated with
# question difficulty are the location of the sub on the numberline
# (ucq below) and the mean reaction time of players encountering that
# question (or maybe the sd, etc. of the same reaction times).  I
# explore these briefly below...

ucq <- as.numeric(names(easiness))
par(mfrow=c(2,3))
plot(prop.correct,mean.reacTime.per.player)
plot(easiness,mean.reacTime.per.question)
plot(ucq,mean.reacTime.per.question)
plot(prop.correct,sd.reacTime.per.player)
plot(easiness,sd.reacTime.per.question)
plot(ucq,sd.reacTime.per.question)

cor(logit(easiness),mean.reacTime.per.question)
cor(logit(easiness),sd.reacTime.per.question)
cor(easiness,mean.reacTime.per.question)
cor(easiness,sd.reacTime.per.question)
cor(logit(easiness),ucq)
cor(easiness,ucq)

summary(lm(easiness ~ ucq))
summary(lm(easiness ~ ucq + I(ucq^2) + I(ucq^3) + I(ucq^4)))
summary(lm(easiness ~ mean.reacTime.per.question))

##############################################################
```