Multilevel Modeling of Categorical Data

Sean Devine

24-03-2021

Preamble

In this tutorial, I want do to four things:

- Demonstrate how to compute and interpret (multilevel) logistic regression models;
- Demonstrate how to create a design matrix for categorical predictor variables and interpret associated coefficients;
- Demonstrate how to check assumptions for logistic multilevel models;
- Apply all of these echniques to data I collected about people's effort preferences. A quick word about notation. Throughout the tutorial, I will use various equations to represent the variables I will be discussing. Greek letters (β, γ, etc.) are used to represent individual parameters, whereas bold Arabic letters ($\mathbf{X}$, $\mathbf{B}$) represent matrices of parameters. We haven't seen in class how to represent regressions or multilevel models in matrix notation, which is unfortunate because it is very useful for representing categorical variables and often comes up in the literature. A full review of matrix notation for MLM is out of the scope of the current tutorial, but for pedagogical purposes, I added a supplement at the end that goes through representing multilevel models this way. This is optional reading for those who are interested and does not really bare on the rest of the tutorial.

Logistic Regressions, Probabilities, and Odds Ratios

Logistic regressions are regression models used to model data when the outcome variable is binary. Binary variables are useful for measuring many variables of interest in psychology: whether a professor has tenure or not, someone's biological sex, and, as we will see, choices in a decision-making experiment. The problem is that the models we have seen in class are *linear* multilevel models. In other words, they are models where you have to be able to draw a line between two points. When we have binary data, this is a problem.

To illustrate this, let's work with the following (made-up) example. Imagine we collect data on whether someone will answer "yes" or "no" to the question: "Will you go on a data with me?" We predict that their answer will depend on how beautiful the person asking is (0 = not beautiful at all to 200 = very beautiful). **Figure 1a** shows what these data might look like[1].

As we can see, this does not look like the typical scatterplot we would get from a linear model, where the relationship between the variable on the x-axis and the y-axis is immediately clear (e.g., when x goes up, so does y). This is because the variable on the y-axis is binary: people can only say "yes" or "no" to a date (here we have dummy coded "yes" and "no" as 1 or 0, respectively). Each point represents one answer.

While maybe a little foreign, we can still learn something from this figure. Namely, we can see that there are more "no" points when beauty is low and more "yes" points when beauty is high. Naively, we might conclude something like: "people say yes to dates more often when the person asking is more beautiful". This is an intuitive conclusion to reach, but the goal of statistical modeling is that we can make more nuanced claims

[1] These data are simulated. Hopefully people are not this shallow in real life.

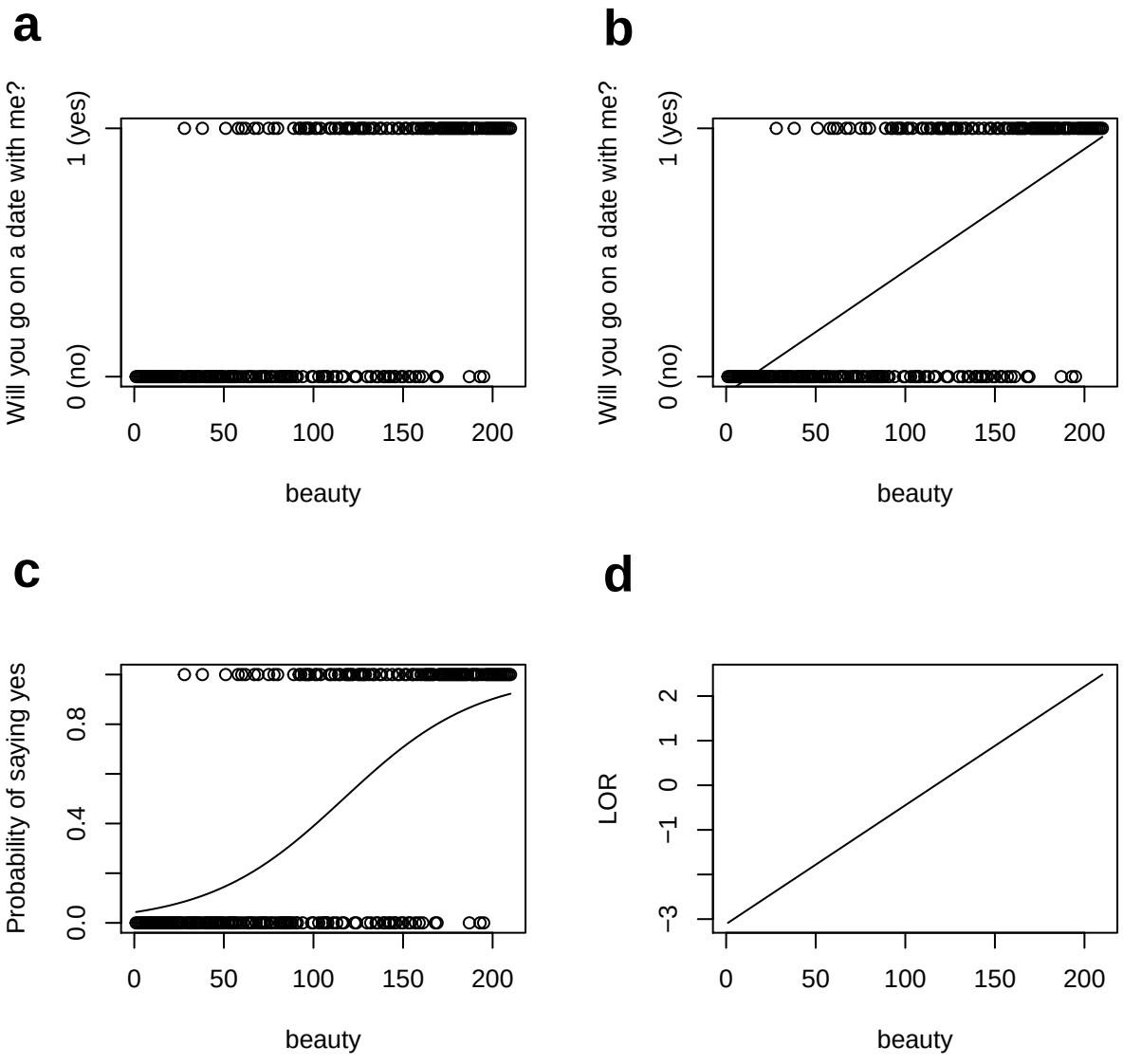


Figure 1: From binary data to linear models.

about the relationship between beauty and likelihood of getting a date than just the raw frequencies allow us to.

If we model these data in the traditional way, using a linear model, we would get something like **Figure 1b**. Here, the values on the regression line are computed the same way as usual for a level-1 model:

$$Date_i = \beta_0 + \beta_1 Beauty_i \tag{1}$$

At first glance, this approach might seem reasonable: better-looking people get more dates. There are two problems, however. First, some values on the regression line are impossible. What would it mean to be halfway between "yes" and "no"? Second, this model predicts that some of the values will exceed 0 or 1 in either direction, which doesn't make sense: how can we say "more than yes" to a date?

Still, there is something intuitive about the linear approach. In fact, if we changed the y-axis so that these impossible values went away, we would have something that makes sense. To solve this problem, we can estimate the *probability* of someone saying yes to a date, rather than the binary answer. This would fixe both our problems, because probabilities are bound from 0 to 1 and are interpretable in a continuous manner. The question is how do we move from binary data to probabalistic data? Fortunately, we can use a *logistic* function to do so:

$$logistic(\eta) = \frac{1}{1 + e^{-\eta}}. \tag{2}$$

The logistic function[2] squeezes the output from our linear model to be between between 0 and 1, which we use to represent probabilities of responding. As an input, it takes η, which in our regression framework is the predicted value of y from our linear model (i.e., the result of $\beta_0 + \beta_1 Beauty_i$). We can see what this function looks like for our data in **Figure 1c**. The characteristic *S-curve* better represents our earlier intuition: the better-looking someone is, the higher probability with which they'll get a date.

While transforming our data using a logistic function fixes our the issue of impossible values on our regression line, we have lost the nice linear function we had in **Figure 1b**. As such, we cannot apply the (multilevel) *linear* techniques we have learned in class and previously in our statistical training. To move from this non-linear logistic function to a linear one, we have to "invert" this logistic function to recover the linear relationship we lost. In other words, we want to get back to something like $Date_i = \beta_0 + \beta_1 Beauty_i$, while keeping interpretable values for every predicted value of $Date_i$ (i.e., keeping probabilities as our y-axis). To do so, we can use the following equation.

$$log\left(\frac{p(y_i = 1)}{p(1 - y_i = 1)}\right) = \beta_0 + \beta_1 Beauty_i \tag{3}$$

Now, our outcome is no longer the probability that $y_i = 1$, but instead the the **log odds of y_i being equal to 1** (which in this case means saying "yes" to a date). This is simply the natural logarithm of the probability that $y_i = 1$ (which we got from our sigmoidal function earlier) over the probability that $y_i \neq 1$. Therefore, we can easily move from log odds (LOR) to regular odds ratios (OR) to probability as follows:

$$\begin{aligned} LOR &= log\left(\frac{p(y_i = 1)}{p(1 - y_i = 1)}\right) \\ OR &= LOR^e \\ p(y_i = 1) &= \frac{OR}{1 + OR}. \end{aligned} \tag{4}$$

In other words, when $LOR = 0$, $OR = 1$, $p(\mathbf{Y}_i = 1) = 0.5$. The change in LOR will be the regression coefficients we get from a logistic regression model. In the next section, I will explain how to run a logistic

[2] e here is Euleur's constant, which is approximately equal to 2.718282

regression model with multilevel data in R and we will interpret the resultant coefficients using the equations from (3)

Data Demo

With the theory out of the way, we can now focus on actually modeling multilevel binary data using logistic models.

Description of the Data

The data for this tutorial comes from an experiment I ran earlier this year through Amazon Mechnical Turk (AMT). The data stored in `effort_progress_choice.csv` describes 60 participants' choices in a cognitive experiment about mental effort preference when progress feedback is present or not.

Study Background

Past work has shown that people find cognitive effort aversive and,if given the option, they will avoid it (Kool et al., 2010). However, some activities (e.g., puzzle-solving, Suduko, building furniture) are performed by humans *because* they are effortful, not despite the fact (Inzlicht et al., 2018). In this study, we wanted to see whether **progress** was something that motivated people to freely choose harder mental work. The main question was whether the presence of progress feedback in a mentally demanding task would make people choose to work harder.

Experimental Task

The experiment consisted of a series of two repeating phases: a choice and judgement phase (see Figure 2).

In the judgement phase, participants were asked to complete a cognitively demanding task: a task-switching task. Here, participants had to judge a series of numeric digits that appeared on the screen one at a time (called probes). These probes could be one of two colours: green or orange. If the probe was green, participants were asked to make a magnitude judgment: whether the numeric probe was greater or smaller than 5. If the probe was orange, participants were asked to make a parity judgment: whether the number is even or odd. In isolation, these tasks are not very hard to perform. However, when interleaved on a trial-by-trial basis, the task becomes significantly more demanding. The rate at which the rules switch (the switch rate) can be thought of as an index for difficulty, with higher switch rates corresponding to more demanding switch tasks for a given series of probes (called a block). For instance, a block of 3 trials where the rules switch from magnitude-parity-magnitude is thought to be mentally more demanding than a block where the rules stay the same: magnitude-magnitude-magnitude. The key manipulation is that the switch rate (i.e., the difficulty) in a given block is controlled by participants' choices in the choice phase.

In the choice phase, participants were presented with two (virtual) decks of cards to choose from. Each deck corresponded to a switch rate that participants would have to complete in the subsequent judgement phase. The three possible switch rates were 10%, 50%, and 90%, from easiest to hardest. In addition, half of these decks were also "progress decks". When a progress deck was chosen, participants received within-block progress feedback on the subsequent switch task (i.e., a green bar filled following each judgement; see **Figure 2**). When a "no-progress" deck was chosen, the switch task in the judgement phase was the same, but they received no progress information.

As a result of this design, there were six total decks (labels from A-F) participants could choose from, varying according to their associated switch rate and whether or not they produced progress feedback.

Table 1: Characteristics of the Decks in the Experiment

	Yields Progress	Does Not Yield Progress
Low Demand	A	D

	Yields Progress	Does Not Yield Progress
Medium Demand	B	E
High Demand	C	F

Overall then, participants were exposed to 15 unique possible deck pairings (A vs. F, B vs. C, etc.), each of which were presented 4 times throughout the experiment. These 15 pairings were fully counterbalanced, allowing for conclusions to be drawn about participants' pure effort preferences (when a deck choice was between two no-progress or progress decks that differed in terms of switch rate), pure progress preferences (when a deck choice was between a no-progress deck and progress deck that had the same switch rate), and the interaction between the two (when a deck choice was between a no-progress deck and a progress deck that varied in terms of switch rate).

Research Question

In summary, what we are trying to answer is whether people are willing to work mentally harder when that mental effort explicitly conveys progress information. To wit, the key variables of interest for this tutorial are participants' choices (high/low demand deck; progress/no prorgess deck) and how these choices varied as a function of what deck pairings they were choosen from. Specifically, we will try to model whether participants' were more likely to choose one type of deck over another depending on (a) the demand pairing (how hard each available deck was) and (b) the progress pairing (whether one, both, or none of the decks gave progress feedback) on a given trial.

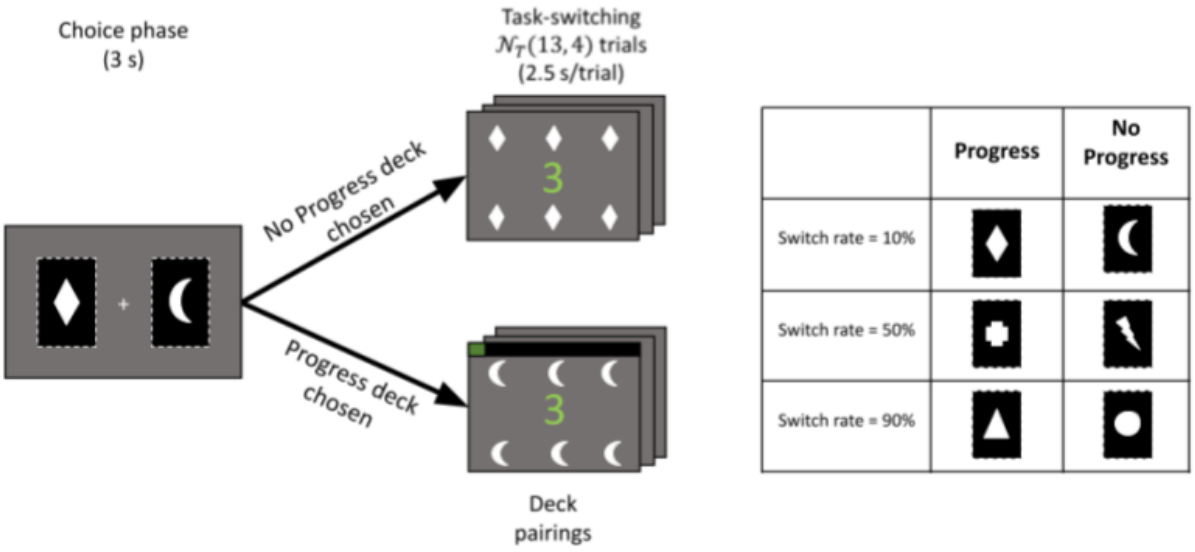


Figure 2: Task Description

Hypotheses

We had three hypotheses before collecting the data:

- **H1**: There would be a main effect of demand pairing (IV) on effort choices (DV), such that, holding progress constant, people would select low demand decks more often than chance.
- **H2**: There would be a main effect of progress pairing (IV) on effort choices (DV), such that, holding demand constant, people would select progress decks more often than chance.

- **H3**: There would be an interaction between progress pairing (IV1) and effort pairing (IV2) on effort choices (DV), such that people would select high demand decks more often than chance when they confered progress and less often when they didn't.

Data Preperation and Organization

As you may have gathered from the previous section, all the data in this tutorial are *categorical*. The outcome (participants' choices) are binary (high effort/low effort; progress/no progress) and the predictors are the different types of deck pairings someone could choose from (e.g., deck A vs. D, as shown in **Figure 2**). So what do these data look like and how do we model them?

First, we need to load the data.

```
data <- read.csv('effort_progress_choice.csv') # load the data
head(data)                                     # take a peak
```

```
##               id decisionnum deckl deckr pair deckchoice effort_type
## 1 A11YS0T8MV3Q7C           1 deck5 deck1  1-5         76          LM
## 2 A11YS0T8MV3Q7C           2 deck5 deck3  3-5         82          MH
## 3 A11YS0T8MV3Q7C           3 deck1 deck4  1-4         82          LL
## 4 A11YS0T8MV3Q7C           4 deck2 deck4  2-4         82          LM
## 5 A11YS0T8MV3Q7C           5 deck4 deck3  3-4         82          LH
## 6 A11YS0T8MV3Q7C           6 deck4 deck6  4-6         76          LH
##     prog_type effchoice progchoice
## 1 NoProg_Prog         1          1
## 2 Prog_NoProg         1          0
## 3 NoProg_Prog        NA          1
## 4 Prog_NoProg         0          1
## 5 Prog_NoProg         1          0
## 6   Prog_Prog         0          1
```

- `id` is the participant's unique ID number;
- `decisionnum` is which decision that deck choice was for the participant (the first, the second, etc.);
- `deckl` is the deck that shows up on the left of the screen;
- `deckr` is the deck that shows up on the right;
- `pair` is the unique pairing, regardless of which side the decks showed up on (there are of 15 these);
- `deckchoice` is which deck participants' chose (the deck on the left = 76; right = 82);
- `effort_type` is the demand pairing for those decks:
  - `LL` = Both decks are low demand (both lead to a switch task with a 10% switch rate);
  - `LM` = One deck is low demand and the other is medium demand (50% switch rate);
  - `LH` = One deck is low demand and the other is high demand (90% switch rate);
  - `MM` = Both decks are medium demand;
  - `MH` = One deck is medium demand, the other is high demand;
  - `HH` = Both decks are high demand;
- `prog_type` is the progress pairing for those decks:
  - `NoProg_NoProg` = Neither deck yields progress feedback on the switch task
  - `Prog_Prog` = Both decks yield progress feedback on the switch task;
  - `NoProg_Prog` = The high demand deck yields progress feedback;
  - `Prog_NoProg` = The low demand deck yields progress feedback;
- `effchoice` is whether or not participants chose the highest demand deck available that trial (1 = yes, 0 = no, `NA` = both decks had the same demand level);

- `progchoice` is whether or not participants chose the deck that yielded progress feedback that trial (1 = yes, 0 = no);

This may seem like a lot of information, but really all we are trying to do is predict `effchoice` and `progchoice` from `effort_type` and `prog_type`. From this description, it should be clear that we need a multilevel to do so, because `effchoice` and `progchoice` are both nested within participants — that is, each participant made 60 choices.

Defining the Design Matrix

Now that we have a sense of our data, we have to determine how to model it. Namely, we have to find a way to quantify all of these categorical predictors into numeric variables that can be included in the model. In other words, we need something that can be multiplied by our coefficients, γ_{xx}.

As we learned earlier in the semester, we can assign numeric labels to categorical predictors by using a *coding scheme*. In our case, we are interested in choices within a certain deck pairing and less interested in how these deck pairings differ from each other. For this reason, we will use a **dummy coding scheme**, where a value of 1 represents that level of the category (that deck pairing) is present and 0 represents when it is not. R does this for us automatically when our variable is a factor. We can access this default dummy coding scheme using by using the contrast function.

```
contrasts(data$effort_type) # get dummy codes for effort_type
```

```
##    LH LL LM MH MM
## HH  0  0  0  0  0
## LH  1  0  0  0  0
## LL  0  1  0  0  0
## LM  0  0  1  0  0
## MH  0  0  0  1  0
## MM  0  0  0  0  1
```

```
contrasts(data$prog_type)   # get dummy codes for prog_type
```

```
##                NoProg_Prog Prog_NoProg Prog_Prog
## NoProg_NoProg            0           0         0
## NoProg_Prog              1           0         0
## Prog_NoProg              0           1         0
## Prog_Prog                0           0         1
```

From these contrasts, we can see that `HH` and `NoProg_NoProg` are our intercepts (the values when our dummy codes are all equal to zero). When we apply dummy coding in a regression model, the regression coefficients will reflect the difference in our outcome between a particular level of our predictor and this intercept. For theoretical reasons, we want to change the reference level for the `effort_type` variable to reflect the `LH` level–the comparison between the easiest and hardest decks. We do so because past work has used similar manipulations and have focused specifically on this pairing, so insofar as we are interested in differences between the pairs, we should compare to this one (Kool et al., 2010). Changing the coding scheme is easy in R:

```
# The relevel command reconstructs the dummy coding matrix so that the chosen "ref" level is the interc
data$effort_type <- relevel(data$effort_type, ref = 'LH')
contrasts(data$effort_type) # let's take a look now
```

```
##    HH LL LM MH MM
## LH  0  0  0  0  0
## HH  1  0  0  0  0
## LL  0  1  0  0  0
## LM  0  0  1  0  0
## MH  0  0  0  1  0
```

```
## MM  0  0  0  0  1
```

Now we can see that the intercept for `effort_type` is `LH` and intercept for `prog_type` remains `NoProg_NoProg`.

Model Specification and Hypothesis Testing

Now that we have quantified our categorical data using dummy codes, we can fit the multilevel models. Specifically, we will estimate the probability of choosing the high demand deck or the progress deck in each of the deck pairings that address our hypotheses. We will do this for each of our hypotheses.

H1

First we will test people's effort preferences when progress is held constant. That is, we will see if people prefer high or low demand decks when both decks yield progress or neither deck yields progress.

To do so, we first need to subset the data so as to only include trials that are relevant to H1: those where progress was held constant (where both decks either did or did not yield progress).

```
# 1. Subset the data to only include decisions relevant to H1.
h1.data <- data[data$prog_type %in% c('Prog_Prog', 'NoProg_NoProg'),]
```

To actually fit the model is relatively straightforward now, since the parameter of interest for H1 is $\gamma_0 0$: the overall log odds of participants' preferring low demand decks when progress is held constant. Thus, we can fit a null model and address our hypothesis.

$$log\left(\frac{p(\text{effchoice}_{trial,id}=1)}{1-p(\text{effchoice}_{trial,id}=1)}\right)=\gamma_{00}+U_{0id}+R_{trial,id} \tag{5}$$

```
# 2. Formulate a null model
# first load lme4
library(lme4)

# fit the null model:
h1.0 <- glmer(effchoice ~ 1 + (1|id), data=h1.data, family = 'binomial')
summary(h1.0)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: effchoice ~ 1 + (1 | id)
##    Data: h1.data
##
##      AIC      BIC   logLik deviance df.resid
##   1931.0   1941.5   -963.5   1927.0     1438
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -1.070 -0.830 -0.666  1.106  1.728
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  id     (Intercept) 0.1703   0.4126
## Number of obs: 1440, groups:  id, 60
##
## Fixed effects:
```

```
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.40840    0.07666  -5.327 9.96e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note here that to fit a logistic model with `lme4`, all you need to do is: * add a "g" in from of `lmer`. The g stands for "generalized", as in "generalized linear model"; * make sure to include `family = 'binomial'`, which tells the model that the outcome here is binary.

Conveniently, if either of these steps are skipped, `lme4` will throw up a warning so you don't forget!

Now to the output. We can begin by interpreting the fixed effects.

Remember, γ_{00} here represents the overall log odds of choosing the highest demand deck across participants. Using the equations we saw earlier, we can convert these log odds to probabilities:

$$OR = LOR^e$$
$$p(y_{ij} = 1) = \frac{OR}{1 + OR}.$$

In R we can compute this like this:

```
G00 = summary(h1.0)$coefficients[,'Estimate'] # this store G00
OR = exp(G00)                                 # exp() raises whatever number to the power of e
pY_is_1 = OR/(1+OR)
pY_is_1
```

```
## [1] 0.3992957
```

This is telling us that the model predicts that, on average, people chose the high demand task about 40% of the time when progress was held constant. To be sure that we didn't make a mistake, we can just check the same probability in our empirical data.

```
# Because G00 is the biggie intercept, all we have to do is take the gloabl mean.
mean(h1.data$effchoice)
```

```
## [1] 0.4034722
```

As we can see, the model's predicted value and the empirical probability of choosing the high demand task are very close: about 40% in both cases.

Looking back to our model output, we can also see that γ_{00} is significantly different from zero according a Wald test[3]. Recall, $LOR = 0$ is the same as $OR = 1$ or $p(y_{ij}) = 0.5$. Therefore, this is the same as saying that participants chose high demand decks significantly less than chance (i.e., 50% of the time). To confirm this intuition, we can also check the profile confidence intervals:

```
h1.0.ci <- confint(h1.0, oldNames=F)
h1.0.ci
```

```
##                          2.5 %     97.5 %
## sd_(Intercept)|id   0.2547739  0.5890846
## (Intercept)        -0.5640924 -0.2576024
```

Here we can see that the confidence interval for the log odds of choosing the high demand task does not cross zero: 95% CI = $[-0.56, -0.26]$. Applying the same computations as earlier, this is the same as saying

[3] `lme4` outputs Wald z-statistics for coefficients because they are easy to compute. However, they also rely on some sometimes suspect assumptions. I will cover assumptions of logistic multilevel models soon, but for now I point interested readers to this resource: https://bbolker.github.io/mixedmodels-misc/glmmFAQ.html#what-are-the-p-values-listed-by-summaryglmerfit-etc.-are-they-reliable. In general, this website is a useful resource for anyone fitting logistic multilevel models

that the CI for choosing high demand decks overall is $[36\%, 44\%]$. These results support our first hypothesis and show that, all else being equal, people avoid high cognitive effort tasks more than we would expect from chance (50% responding). We can see a summary of these results in **Figure 3**.

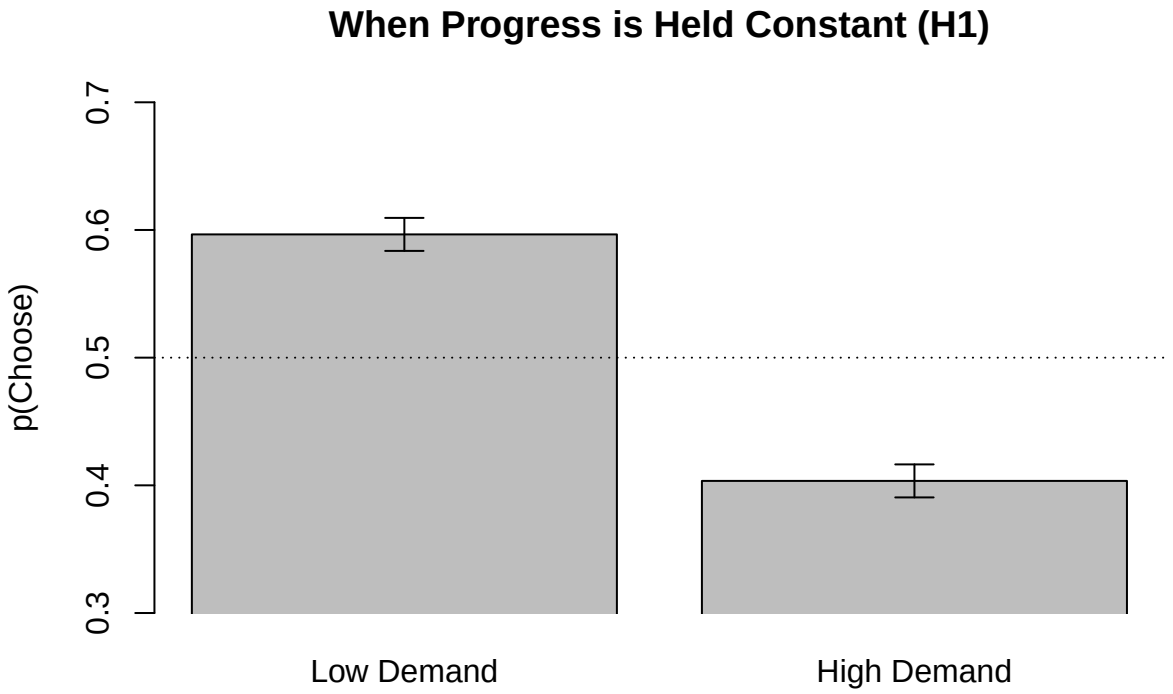


Figure 3: Results for H1

Interestingly, we also see from this output that the CI for τ_0^2 is not bound at zero, suggesting that there is significant variation between participants in how much they avoid the high demand decks overall, when progress is held constant. To quantify the degree of variation between participants in a single number, we can compute the intraclass correlation coeficient (ICC).

In the context of logistic regression however, we *cannot* compute the ICC in the normal way, as $\frac{\tau_0^2}{\tau_0^2+\sigma^2}$. Normally σ^2 represents the level-1 residuals: how much a predicted observation $\hat{y}_i$ differs from a participant j's observed y_i. In a logistic framework however, the residuals have a different interpretation. I will discuss the residual structure of logistic models later on, but for now the key point to keep in mind is that the residuals in these models *depend on* the outcome in a way that they don't in linear models. No independent level-1 residuals means no σ^2, whicih means no immediately accessible ICC.

There are a few statistical solutions to this problem [4]. The first is quick heuristic one where we approximate $sigma^2$ by substuting it for the known variance of the *logistic distribution*. This is known as the *latent variable approach*.

$$ICC \text{ (aka } \rho) = \frac{\tau^2}{\tau^2 + \frac{\pi^2}{3}}. \tag{6}$$

[4] see this blogpost for a good introduction and additional resources: https://www.barelysignificant.com/post/icc/. For a more scholarly source, see Goldstein & Rasbash (1996) and Goldstein, Browne, and Rashabch (2002)

```r
tau2 <- summary(h1.0)$varcor$id[1] # this extracts tau2 from the model
h1.0.icc <- tau2/(tau2 + pi^2/3)   # this computes the icc according to eq. above
h1.0.icc
```

```
## [1] 0.04920937
```

In other words, about 5% of the variance in participants' overall effort preferences stems from individual differences between participants. Importantly, this method assumes that our outcome variable represents a dichotomous "cut-off" of some underlying continuous variable. For example, "healthy" or "sick" are cutoffs for otherwise continuous measures of health. This assuption is what allows us to use the variance from the (continuous) logistic distribution in place of σ^2.

An approximation method that is more agnostic towards the theory underlying our data is a *simulation-based approach*. Here, we use the estimates from our model to simulate a large number of expected level-1 probabilities, using τ_0^2 as a "hyperparameter" (basically the standard deviation for our simulated distribution). With this, we compute a Bernouilli variance for each simulated probability: $var_{ij} = p_{ij}(1 - p_i j)$. We then take the mean of these variances and compute the ICC as follows:

$$\rho = \frac{Var(p_{sim})}{Var(p_{sim}) + \frac{1}{N}\sum var_{sim}} \tag{7}$$

```r
N <- 1e5 # number of simulations (some big number)

# 1. Sample a bunch of G00 from a normal distribution with mean = G00 and SD = tau2_0
G00_sim <- rnorm(N, mean=G00, sd=tau2)

# 2. Turn these G00_sim into probabilities using our equation:
# p(y_ij=1)_sim <- exp(G00_sim)/(1+exp(G00_sim))
p_sim <- exp(G00_sim)/(1+exp(G00_sim))

# 3. Compute the Bernouilli variance for each of these probabilities
v_sim <- p_sim * (1-p_sim)

# 4. Take the mean of these Bernouilli variances
meanv_sim <- mean(v_sim)

# 5. Compute ICC as above
ICC <- var(p_sim)/(var(p_sim) + meanv_sim)
ICC
```

```
## [1] 0.006864859
```

Applying this technique we get a much lower estimate of the ICC, where the variation between participants' effort preferences when progress is held constant is now only about 1%, down from 5% with the previous technique. A downside to this approach is that the ICC is always conditional. In our case, we had no predictor variables other than the intercept, γ_{00}, but if we did have predictors we would need to include them and their dummy coding in step 2 of the R code. This means that the ICC is always conditional upon the included covariates (which may be of interest to you, of course!)

It's difficult to judge which of these approaches are best overall and will come down to the assumptions of your data set and what you are aiming to test. In the current data, we can look at the variation in γ_{00} directly from the data. These are summarised in **Figure 4** for a better sense of theparticipant-level differences in effort preference when progress is held constant

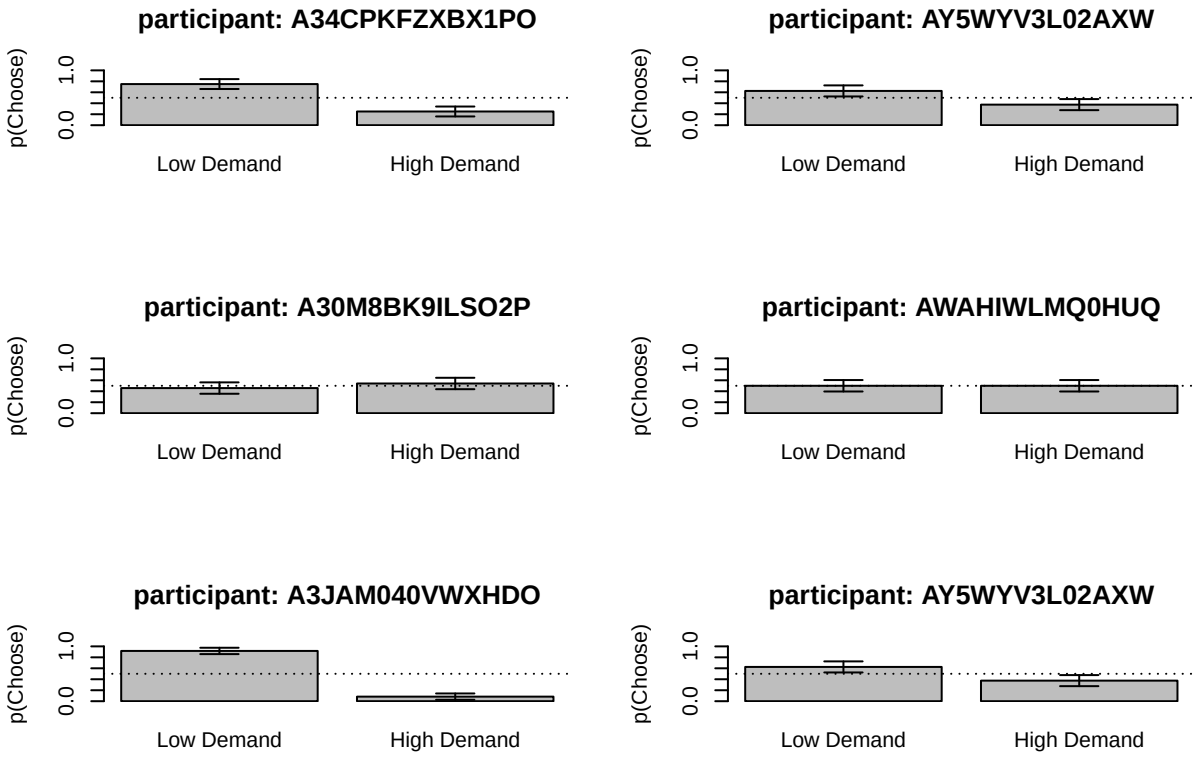


Figure 4: Individual variation for 6 participants in demand preference, when progress is held constant (H1)

H2

We can test H2 using much of the same techniques we did for H1. Namely, we will do the following: * Subset the data to select for trials where demand is held constant (people choose only between whether they prefer progress or no progress); * Fit a model that predicts people's progress preferences (`progchoice`) from γ_{00}; * Compute the ICC; * Interpret the output.

$$log\left(\frac{p(\text{progchoice}_{trial,id}=1)}{1-p(\text{progchoice}_{trial,id}=1)}\right)=\gamma_{00}+U_{0id}+R_{trial,id} \tag{8}$$

```
# 1. Subset data
# only include trials where demand is the same (so effchoice is NA)
h2.data <- data[is.na(data$effchoice), ]

# Fit the model
h2.0 <- glmer(progchoice ~ 1 + (1|id), data=h2.data, family='binomial')
summary(h2.0)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: progchoice ~ 1 + (1 | id)
##    Data: h2.data
##
##      AIC      BIC   logLik deviance df.resid
##    954.9    964.1   -475.5    950.9      718
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.6852 -1.1523  0.6424  0.8064  1.0034
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  id     (Intercept) 0.2559   0.5059
## Number of obs: 720, groups:  id, 60
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.4998     0.1033    4.84  1.3e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here, we see that across demand levels, the log odds for people prefering progress decks to non-progress decks is positive and significant. Using the same steps as earlier we can convert these log odds to a probability and check how closely they match up to what we observe in the data.

```
# Convert log odds to probabilities
G00 = summary(h2.0)$coefficients[,'Estimate'] # this store G00
OR = exp(G00)
pY_is_1 = OR/(1+OR)
paste0('estimated p_ij = ', pY_is_1)
```

```
## [1] "estimated p_ij = 0.622412895658938"
```

```
# Check empirical probability
paste0('empirical p_ij = ', mean(h2.data$progchoice))
```

```
## [1] "empirical p_ij = 0.615277777777778"
```

Again, these match up very closely and tell us that across demand levels, people selected progress decks about 62% of the time and this was significantly more than we would expect from chance (50%). To double-check this significance test, we can compute profile confidence intervals again.

```
h2.0.ci <- confint(h2.0, oldNames=F)
```

```
## Computing profile confidence intervals ...
```

```
h2.0.ci
```

```
##                          2.5 %    97.5 %
## sd_(Intercept)|id 0.2518917 0.7707543
## (Intercept)       0.2987586 0.7131055
```

Here too, we see that the 95% CI for γ_{00} does not cross 0 and, in probability terms, predicts with 95% confidence that people's true progress preference in the population is within [57%, 67%]. These results support our second hypothesis and show that, all else being equal, people prefer progress information. These data are summarised in **Figure 5**.

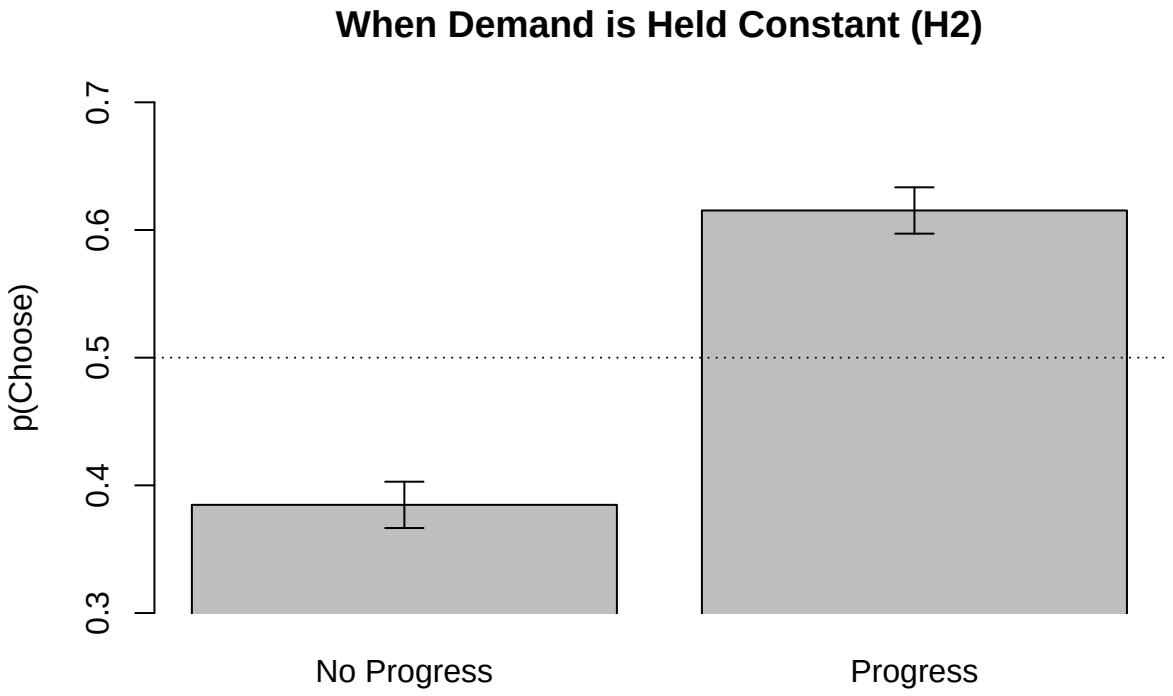


Figure 5: Results for H2

We again notice significant variation in this preference between individuals, as indexed by the 95% CI for τ_0^2 not being bounded at 0. To quantify this, we compute the ICC using the latent variable approach, but interested readers are welcome to try the simulation approach as well.

```
tau2 <- summary(h2.0)$varcor$id[1]
h2.0.icc <- tau2/(tau2 + pi^2/3)
```

```
h2.0.icc
```

```
## [1] 0.07218237
```

This tells us that about 7% of the variance in participants' progress preferences is explained by individual differences that exist between participants. As before, we can get an intuition for this by simply looking at the data for different participants when demand is held constant (see **Figure 6**).

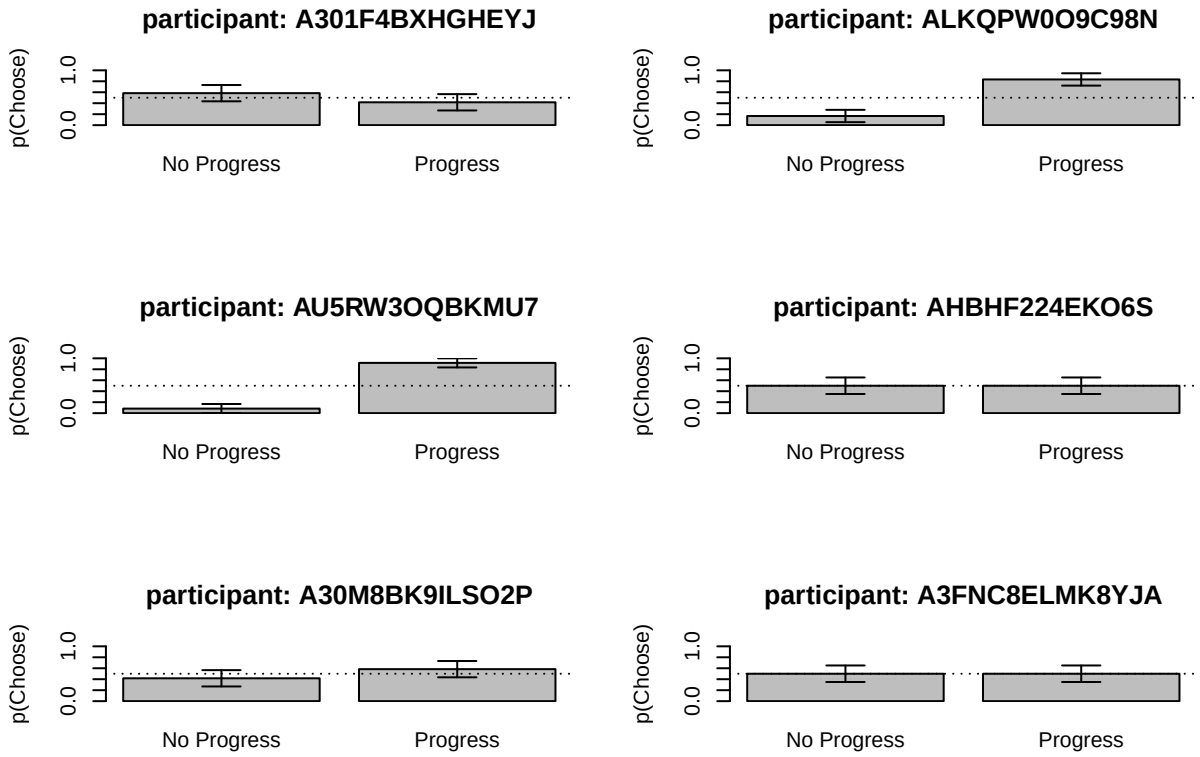


Figure 6: Individual variation for 6 participants in progress preference, when demand is held constant (H2)

H3

Finally, we can test H3. To do so, we can use a similar approach to above, but will require more models than before to arrive at a conclusion. Specifically, we want to test the hypothesis when both demand and progress options vary, `effchoice` depends on the interaction between progress pairing and demand pairing. First, let's subset the data.

```
# 1. Subset the data
# We include trials where progress and demand vary and we exclude cases where demand is held constant (
h3.data <- data[data$prog_type %in% c('NoProg_Prog', 'Prog_NoProg') & !is.na(data$effchoice), ]
```

Because the labels are tricky, remember that `NoProg_Prog` refers to the situation where the progress deck is also the harder of the two and `Prog_NoProg` refers to when the no progress deck is the harder of the two. The trick here is to remember that whatever comes after the `_` in the variable name is the type of deck that is the harder of the two.

With our subsetted data on hand, we can fit a null model as before.

$$log\left(\frac{p(\text{effchoice}_{trial,id}=1)}{1-p(\text{effchoice}_{trial,id}=1)}\right)=\gamma_{00}+U_{0id}+R_{trial,id} \tag{9}$$

```
# 2. Fit the null model
h3.0 <- glmer(effchoice ~ 1 + (1|id), data=h3.data, family='binomial')
summary(h3.0)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: effchoice ~ 1 + (1 | id)
##    Data: h3.data
##
##      AIC      BIC   logLik deviance df.resid
##   1958.6   1969.1   -977.3   1954.6     1438
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.0267 -0.8710 -0.7581  1.1169  1.4357
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  id     (Intercept) 0.08149  0.2855
## Number of obs: 1440, groups:  id, 60
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.32888    0.06541  -5.028 4.97e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here we see a similar effect as before: overall people are effort averse and significantly prefer the low demand task ($p(effchoice=1)=0.42$).However, this is not enough to address H3 on its own. What we are interested in is whether this overall demand avoidance is moderated by `effort_type` and `prog_type`. To test this, we need to fit additional models to test the significance of these predictors. The next model we need to consider is one that predicts the likelihood of choosing a high demand task from the demand pairing[5].

$$\mathbf{Y}=\mathbf{XB}+\mathbf{Zu}+\mathbf{RX}=\mathbf{Z}=\underbrace{\begin{bmatrix}0&0&1&0&0\\0&0&0&1&0\\0&0&1&0&0\\&&\cdots&&\end{bmatrix}}_{\text{effort\_type dummy codes}} \tag{10}$$

```
# Fit model with effort_type as predictor
h3.1 <- glmer(effchoice ~ 0 + effort_type + (1|id), data=h3.data, family='binomial')
```

Notice that in this code we have 0 in the fixed-effects portion formula. Without this zero, the intercept of our model will be the dummy-coded intercept, which we specified to be `LH`. If this were the case, all the resultant coefficients would be compared to this level of `effort_type`. This isn't what we want. Instead, what we want is to treat every coefficient *"like an intercept"*, where the estimates and significance tests reflect when these coefficients differ from zero (i.e., chance responding, because $LOR=0=OR=1=p_{ij}=0.5$). Adding the 0 in the formula accomplishes this by telling `glmer` to "delete" the intercept and treat each coefficient as its own intercept, where the contrast is between that coefficient and zero. It's important to note however

[5]If this matrix notation is confusing to you, see the bonus material.

that this **only applies to the fixed effects** and that the random intercepts with variance τ_0^2 will still use the dummy coded reference point. In other words, the U_{0j} will be the deviations from γ_{00}, as if γ_{00} referred to `LH`, as in our original dummy coding scheme. This is fine for our purposes, because we are less interested in this variation. Even if we were, this would be an interesting variable, since past work has focused on this pairing before (Kool et al., 2010).

Before looking into any effects of this model, we first need to see whether the inclusion of `effort_type` improved the model's fit. That is, whether there is a significant overall effect of `effort_type` on `effchoice`. To do so, we will begin by using a likelihood ratio test (LRT).

```
# LRT on the null vs. effort_type model
anova(h3.1, h3.0)
```

```
## Data: h3.data
## Models:
## h3.0: effchoice ~ 1 + (1 | id)
## h3.1: effchoice ~ 0 + effort_type + (1 | id)
##      npar    AIC    BIC  logLik deviance Chisq Df Pr(>Chisq)
## h3.0    2 1958.6 1969.1 -977.29   1954.6
## h3.1    4 1958.5 1979.5 -975.22   1950.5 4.141  2     0.1261
```

Firstly, notice how we did not have to specify or refit the model with `REML = F`. This is because it is not possible to use restricted maximum likelihood to estimate logistic multilevel models; it is by default, and must be, maximum likelihood. Provided that models are nested, this also means we can conduct a LRT without any refitting.

The LRT suggests that there is no overall effect of `effort_type` on `effchoice`: people are equally demand avoidant in all demand pairings, compared to the null model.

As previously mentioned, logistic models do not have an independent residual variance term (σ^2). As such we cannot compute a pseudo-R^2 value to back up this finding. Similarly, because all our predictors are categorical, profile CI won't help us either, because they will not provide a test for the overall effect of `effort_type`. Alternatively, we can turn to parametric bootstrapping, which can be easily implemented using the `pbkrtest` package.

```
# Load the package
library(pbkrtest)

# Use the PBmodcomp function in the same way you would the anova function
# This can take awhile; for now we will set nsim to a smaller number to get a faster, but less precise,
h3.1.0.pb <- PBmodcomp(h3.1, h3.0, nsim=150)
h3.1.0.pb
```

```
## Parametric bootstrap test; time: 151.06 sec; samples: 150 extremes: 16;
## large : effchoice ~ 0 + effort_type + (1 | id)
## small : effchoice ~ 1 + (1 | id)
##         stat df p.value
## LRT    4.141  2  0.1261
## PBtest 4.141     0.1126
```

What we have just done is repeatedly simulate data from `h3.1` and `h3.0`, compute the differences between the deviance of models for each simulated data set, and compare this null distribution to the observed deviance difference. This is computationally intensive, but can give us a second piece of evidence to support the conclusion from our LRT, in lieu of the traditional profile CI and pseudo R^2. As we can see, both the bootstrap test and the LRT are in agreement: no significant effect of `effort_type`.

We can use the same procedure as above to test for the effect of `prog_type` on `effchoice`.

```r
# Fit model with prog_type as a predictor
h3.2 <- glmer(effchoice ~ 0 + prog_type + (1|id), data=h3.data, family='binomial')

# Compare model to null using LRT
anova(h3.2, h3.0)
```

```
## Data: h3.data
## Models:
## h3.0: effchoice ~ 1 + (1 | id)
## h3.2: effchoice ~ 0 + prog_type + (1 | id)
##      npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## h3.0    2 1958.6 1969.1 -977.29   1954.6
## h3.2    3 1874.5 1890.3 -934.23   1868.5 86.135  1  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Compare model to null using parametric bootstrapping
h3.2.0.pb <- PBmodcomp(h3.2, h3.0, nsim=150)
h3.2.0.pb
```

```
## Parametric bootstrap test; time: 134.37 sec; samples: 150 extremes: 0;
## large : effchoice ~ 0 + prog_type + (1 | id)
## small : effchoice ~ 1 + (1 | id)
##          stat df   p.value
## LRT    86.135  1 < 2.2e-16 ***
## PBtest 86.135     0.006623 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Look at the model summary
summary(h3.2)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: effchoice ~ 0 + prog_type + (1 | id)
##    Data: h3.data
##
##      AIC      BIC   logLik deviance df.resid
##   1874.5   1890.3   -934.2   1868.5     1437
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.3641 -0.7171 -0.5868  0.9246  1.9544
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  id     (Intercept) 0.1054   0.3246
## Number of obs: 1440, groups:  id, 60
##
## Fixed effects:
##                       Estimate Std. Error z value Pr(>|z|)
## prog_typeNoProg_Prog  0.15959    0.08657   1.844   0.0653 .
## prog_typeProg_NoProg -0.86167    0.09272  -9.293   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             p_NP_P
## prg_typP_NP 0.217
```

We can see that both the LRT and bootstrapping approach tell us that `prog_type` significantly predicts `effchoice`. By looking at the model output, we can see that when the progress decks are also the lower demand decks of the pair (`Prog_NoProg`), people overwhelmingly prefer them ($p(effchoice = 1) = 0.30$), whereas when progress comes at a higher cognitive cost (`NoProg_Prog`), people are statistically indifferent to choosing between high and low demand and our p-value suggests even a marginal trend towards effort preference ($p(effchoice = 1) = 0.54$). These results support H3.

Finally, we can consider the interaction between `effort_type` and `prog_type` on `effchoice`. The process here is largely the same, except now we have an interaction term `effort_type:prog_type`. Note here the `:`. We use `:` instead of `*` because at this level we are interested in participants' choices for each deck pairing relevant to H3 (since each deck pairing has both a value for `effort_type` and `prog_type`). Using the `*` would be making the assumption that we have some "main effect" for `prog_type` and a separate "main effect" for `effort_type`, as well as the interaction. Instead, what we want is people's probability of choosing the high demand deck in each individual deck pairing and whether this probability is significantly different from chance ($LOR = 0$ or $OR = 1$ or $p_{ij} = 0.5$). To fit this model, we do apply the same steps as above.

```
# Fit model with interaction term
h3.3 <- glmer(effchoice ~ 0 + effort_type:prog_type + (1|id), data=h3.data, family='binomial')

# Compare model to null using LRT
anova(h3.3, h3.2)
```

```
## Data: h3.data
## Models:
## h3.2: effchoice ~ 0 + prog_type + (1 | id)
## h3.3: effchoice ~ 0 + effort_type:prog_type + (1 | id)
##      npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## h3.2    3 1874.5 1890.3 -934.23   1868.5
## h3.3    7 1872.2 1909.1 -929.12   1858.2 10.223  4    0.03684 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Compare model to null using parametric bootstrapping
h3.3.2.pb <- PBmodcomp(h3.3, h3.2, nsim=150)
h3.3.2.pb
```

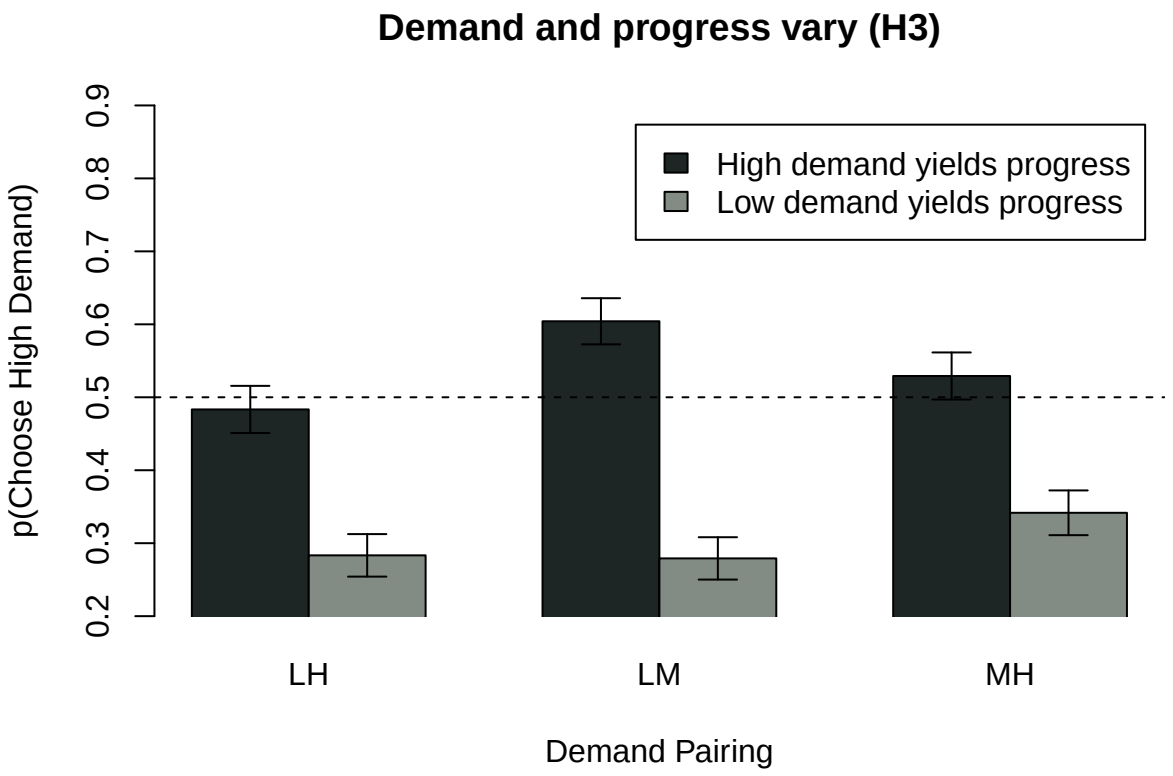
```
## Parametric bootstrap test; time: 683.13 sec; samples: 150 extremes: 1;
## large : effchoice ~ 0 + effort_type:prog_type + (1 | id)
## small : effchoice ~ 0 + prog_type + (1 | id)
##          stat df p.value
## LRT    10.223  4 0.03684 *
## PBtest 10.223    0.01325 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Look at the model summary
summary(h3.3)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
```

```
## Formula: effchoice ~ 0 + effort_type:prog_type + (1 | id)
##    Data: h3.data
##
##      AIC      BIC   logLik deviance df.resid
##   1872.2   1909.1   -929.1   1858.2     1433
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.5725 -0.7634 -0.5794  0.9433  2.0047
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  id     (Intercept) 0.1086   0.3295
## Number of obs: 1440, groups:  id, 60
##
## Fixed effects:
##                                     Estimate Std. Error z value Pr(>|z|)
## effort_typeLH:prog_typeNoProg_Prog -0.06887    0.13763  -0.500  0.61678
## effort_typeLM:prog_typeNoProg_Prog  0.43378    0.14041   3.089  0.00201
## effort_typeMH:prog_typeNoProg_Prog  0.11957    0.13779   0.868  0.38551
## effort_typeLH:prog_typeProg_NoProg -0.95152    0.15127  -6.290 3.17e-10
## effort_typeLM:prog_typeProg_NoProg -0.97256    0.15189  -6.403 1.52e-10
## effort_typeMH:prog_typeProg_NoProg -0.67322    0.14435  -4.664 3.10e-06
##
## effort_typeLH:prog_typeNoProg_Prog
## effort_typeLM:prog_typeNoProg_Prog **
## effort_typeMH:prog_typeNoProg_Prog
## effort_typeLH:prog_typeProg_NoProg ***
## effort_typeLM:prog_typeProg_NoProg ***
## effort_typeMH:prog_typeProg_NoProg ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             e_LH:_N e_LM:_N e_MH:_N e_LH:_P e_LM:_P
## ef_LM:_NP_P 0.093
## ef_MH:_NP_P 0.095   0.094
## ef_LH:_P_NP 0.088   0.083   0.086
## ef_LM:_P_NP 0.087   0.082   0.086   0.084
## ef_MH:_P_NP 0.092   0.087   0.091   0.087   0.087
```

Again, we see good agreement between the LRT and bootstrap. Both suggest that this model outperforms the `prog_type` only model. If we look at the model output, we can see that people significantly avoid demand in `Prog_NoProg` decks and are indifferent or prefer demand in `NoProg_Prog` decks. In particular, we see that the only significant *preference* for demand is in the `LH` and `NoProg_Prog` deck. This makes sense: progress is most valuable for motivating hard mental work in contexts of low demand, where whichever deck someone chooses, they won't work have to work exceptionally hard (either low or medium demand). These results are summarised in **Figure 7**.



Diagnostics for Logistic MLM

With a better sense now of how logistic models are fit and how to interpret them, we can turn to diagnostics. Many of the assumptions are the same as linear MLM, but there are some unique ones to logistic MLM. To illustrate these logistic-specific assumptions, we'll use `h3.3` as an example model.

Level-1 Residuals Are Random

Unlike linear regression, we cannot interpret the level-1 residuals from the model in the same way as usual:

$$r_{ij} = \hat{y}_{ij} - y_{ij} \tag{11}$$

This is because both $\hat{y}_{ij}$ and y_{ij} take on binary values. As such, any histogram or QQ plot would not work; it would yield a bimodal histogram and nonsense QQ plot. To circumvent this, we can think probabilistically: how "off" is the predicted probability of the model from the probability of the observed value. Of course, the probability of the observed value is 1, so what we are really after is a way to quantify the expected probability that $y_{ij} = 0$ or 1 and see how much that *deviates* from what we actually observed. To do so, we can compute a **deviance residual**:

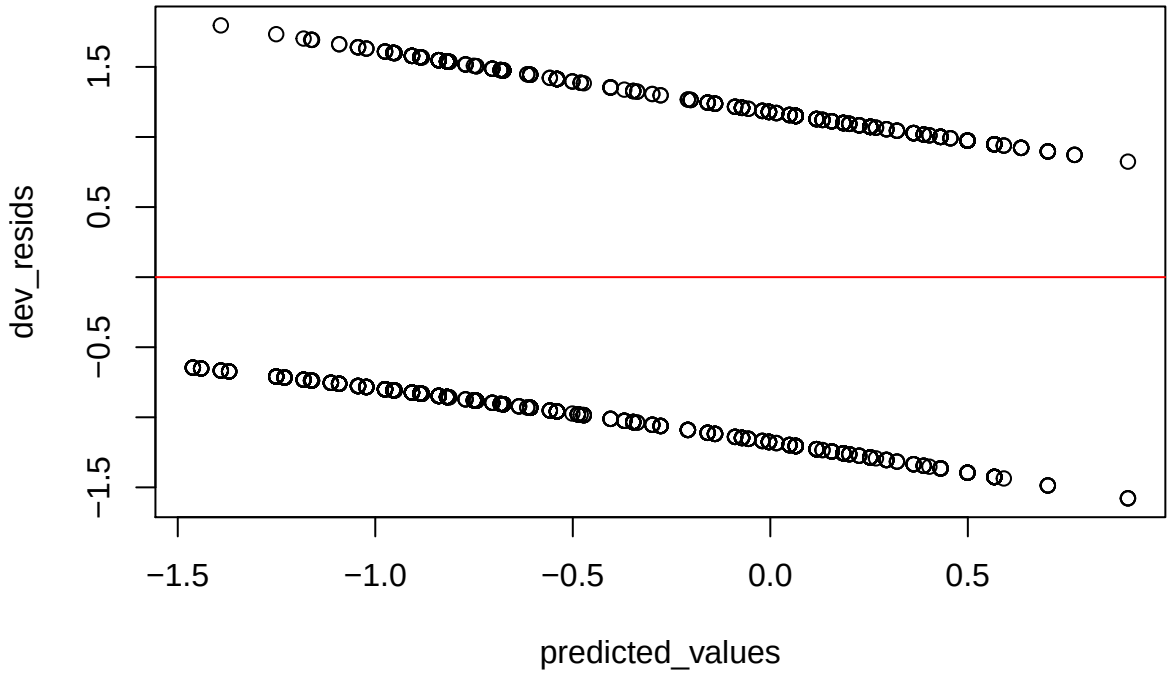
$$r_{ij} = \begin{cases} \sqrt{-2log(\hat{y}_{ij})} & \text{if } y_{ij} = 1 \\ -\sqrt{-2log(1-\hat{y}_{ij})} & \text{if } y_{ij} = 0 \end{cases} \tag{12}$$

$\hat{y}_{ij}$ here is the predicted probability from our model; in other words the estimated likelihood $y_{ij} = 1$. By taking the log of this value, we get the log likelihood. For $y_{ij} = 0$, the interpretation is the same, but we

take $1 - \hat{y}_{ij}$. This is what I meant earlier when I siad the residuals "depend on" the outcome: the value of the residuals depend on the observation.

With these residuals, we can examine the residuals plot just as we would for linear MLM, with one additional caveat. Simply plotting the predicted values against the deviance residuals would yield a plot where the points are scattered equidistant from the zero line, depending on if y_{ij} is 0 or 1, as in this plot:

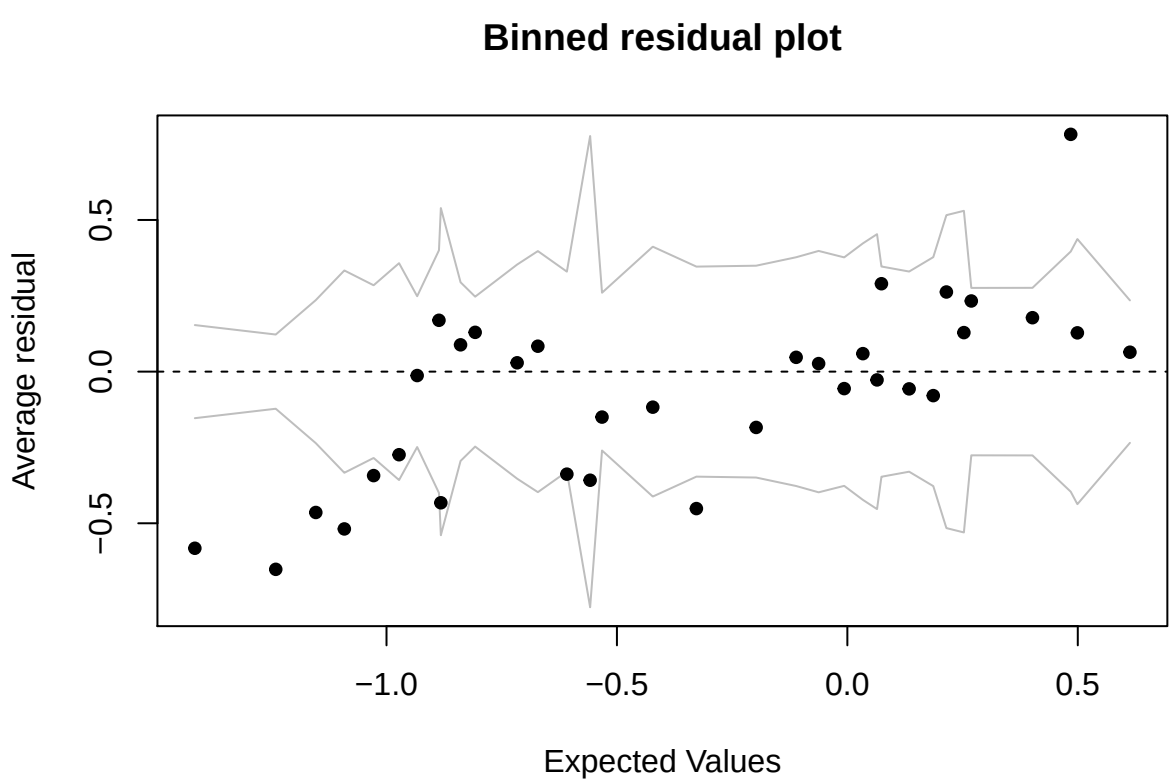
```
predicted_values <- predict(h3.3) # this predicts choice probabilities using the model paramters

dev_resids <- residuals(h3.3, type='deviance') # this computes the deviance residuals as above

plot(predicted_values, dev_resids)
abline(h=0, col='red') # add a zero line in red
```



This isn't very information. We can fix this by binning the residuals by the predicted values and averaging them. This allows us to detect if there is any systematic variation in the residuals acros the predicted values. In other words, if our residuals are truly randomly distributed (as we assumed them to be when fitting the model), we expect that the average of the deviance residuals do not vary systematically according to the predicted values.

We can create such a plot using the `arm` package in R.

```
# Note: don't load the arm library if you use dplyr or other tidyverse libraries, because it can interf
arm::binnedplot(predicted_values, dev_resids)
```



We can read this plot as we would any other residual plot. We expect there to be no systematic variation in the average residuals. The gray lines indicate plus and minus 2 standard-error bounds, within which we expect about 95% of the binned residuals to fall. In our case, this seems to be the case, with maybe some cause for caution for the residuals in the lower expected values. Overall though, the residuals indicate the model is not grossly misspecified.

There is No Over/Underdispersion or Heteroscedasticity Present

A key assumption of any logistic model is that the variance will match the mean of the predicted outcome variable. When using a binomial model, we assume that the variance will be the same as the expected value on the outcome from our model, which we calculate using our regression equation for observation i for participant j. When the variance is larger than predicted by our model, we have **overdispersion**. When it is smaller, we have **underdispersion**.

Over/underdispersion can be caused for a number of reasons. Primarily, it might suggest that the model is misfit in some way, such that the predicted values from the model are not matching up with the actual variation seen in the data. This would sugggest that we are missing some term or interaction that is important. Another issue is known as **zero-inflation**, where the data contains more zeros than the model predicts. A third reason might be that the model is misspecified and we should use some alternative function to lineralize our binomial data (something other than our logistic function).

As in linear MLM, we have to ensure that the variance (or rather, dispersion) in our model is not related to some other variable. In other words, we have to check whether the dispersion is evenly spread across our predictor variables. This is relatively straightforward to do as we will see below.

A powerful package to check both of these assumptions is is the `DHARMa` package. Under this framework, residuals take on a special meaning, distinct from the deviance residuals above. Quoting from the package

vignette[6]:

> "DHARMa aims at solving these problems by creating readily interpretable residuals for generalized linear (mixed) models that are standardized to values between 0 and 1, and that can be interpreted as intuitively as residuals for the linear model. This is achieved by a simulation-based approach, similar to the Bayesian p-value or the parametric bootstrap, that transforms the residuals to a standardized scale. The basic steps are:
>
> 1. Simulate new data from the fitted model for each observation.
> 2. For each observation, calculate the empirical cumulative density function for the simulated observations, which describes the possible values (and their probability) at the predictor combination of the observed value, assuming the fitted model is correct.
> 3. The residual is then defined as the value of the empirical density function at the value of the observed data, so a residual of 0 means that all simulated values are larger than the observed value, and a residual of 0.5 means half of the simulated values are larger than the observed value."

In R, this process is rather simple and it is easiest to explain what we are looking for by simply simulating data using the package. For assumptions testing, we will use the most recent model we fit, namely `h3.3`.

```
# Load the package
library(DHARMa)
```

```
## This is DHARMa 0.3.4. For overview type '?DHARMa'. For recent changes, type news(package = 'DHARMa')
```

```
# 1. Simulate the residuals
h3.3.simulation <- simulateResiduals(h3.3, plot=T)

# 2. Look at the raw residuals if you want
# residuals(h3.3.simulations)

# 3. Plot residuals
plot(h3.3.simulation)
```

On the left, we see a Q-Q plot that we can largely be interpreted the same way we would interpret a regular Q-Q plot. That is, the x-axis shows the model's predicted probabilities at each quantile for $y_{ij} = 1$ and the y-axis shows the same in the observed data. Unsurprisingly, the two values coincide almost identically. This shouldn't be too much of a surprise, since all of our predictors were categorical, and so the model is simply estimating the probability of choosing the harder deck in each of the deck pairings. If you recall from an earlier example we did something similar by hand by computing the model's predicted probability against the empirical mean. Just as in that case, here we see these values match up across quantiles. Indeed, binomial data in *non-clustered* data structures (where we wouldn't need MLM) cannot be under/overdispersed at all (Hilbe, 2013). Nevertheless, in this context, it is always good to check, particularly if your predictors were continuous.

In addition, we also get a series of tests that appear in the margins of the figure. The Kolmogorov-Smirnov (KS) test shows that a non-significant value, which suggests that we should fail to reject the null hypothesis that the predicted values from the model come from a different population than the observed values. We also get a dispersion test, to check for over/underdispersion. The p-value is higher than .05, which suggests that we should fail to reject the null that there is no dispersion problem. Finally, the outlier test shows that we have no significant effects of outliers in the relationship between predicted and observed values.

On the right, we see the scaled residuals across categorical predictors: the deck pairings from `h3.3`. These categorical predictors are assigned numeric labels from 0 to 1, on the x-axis, but I am not sure why. The y-axis are the scaled residuals defined above. A scaled residual of 0.5 means that half of the simulated values were larger than the observed value. As we can see, we have no significant variation in the scaled residuals

[6] https://cran.r-project.org/web/packages/DHARMa/vignettes/DHARMa.html#binomial-data

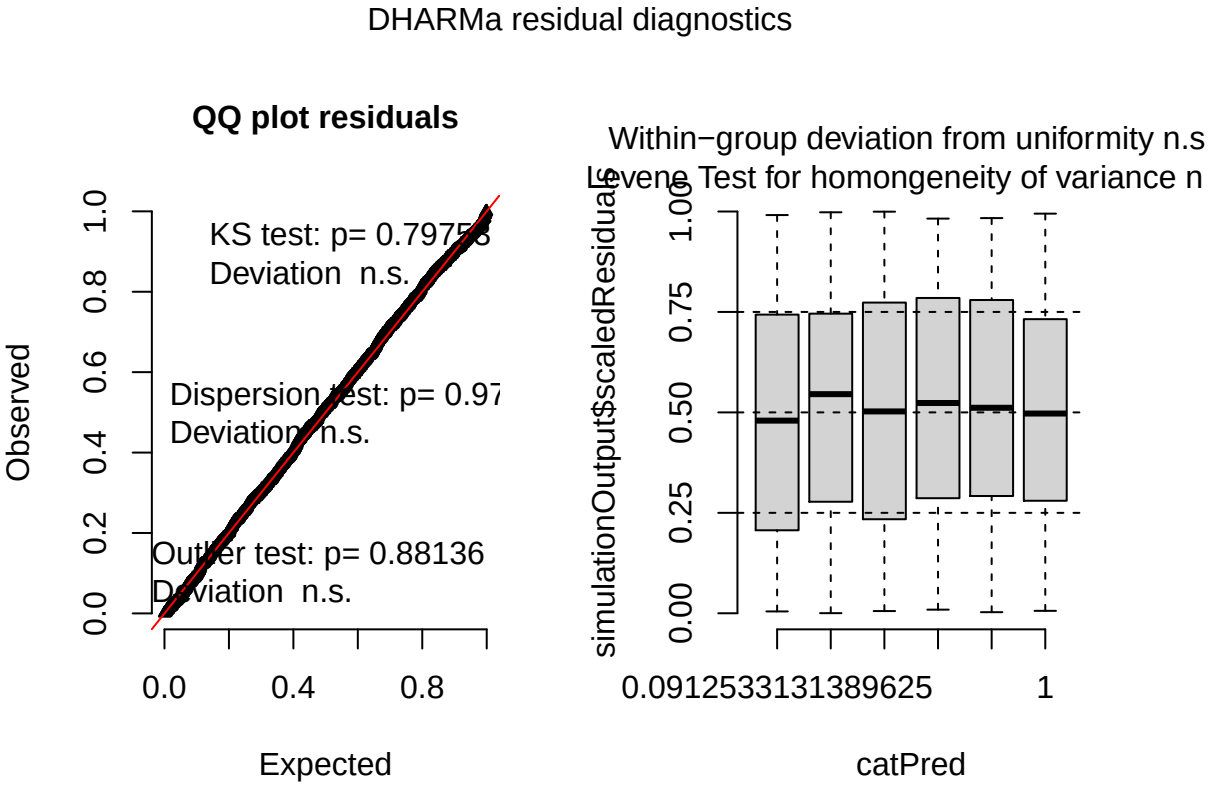


Figure 7: Assumption tests for logistic models with DHARMa: using h3.3

across our categorical predictors and this intuition is backed up by a Levene Test for homogeneity in the title (n.s. stands for non-significant).

Important Note

On the whole, our model seems to satisfy the assumptions for a logistic model.

However, it is important to note that grom what I can gather, specific assumptions for *multilevel* logistic or binomial regressions are actively being researched. I could not find any mentions of assumptions regarding the residual structure between level-1 and level-2, as we saw for linear regression, in specific reference to binomial logistic regression. While checking residuals, overdispersion, and homegeneity of variance are important for any logistic model, it is worth keeping in mind that these are likely not the only factors that might impact your model's fit. As such, this should be considered an **introduction** to these assumptions and not the be-all and end-all of logistic MLM diagnostics.

Conclusion

The goals of this tutorial were to introduce the concepts of logistic multilevel modeling. Hopefully, it has been successful in demonstrating how to implement, interpret, and utilise logistic MLMs for testing your hypotheses.

I did not cover everything you could do with the current dataset, for the sake of space. If anyone is interested in the larger results from the current data, don't hesitate to email me: seandamiandevine@gmail.com!

Bonus Material

Matrix Notation for MLM

In class, we learned that a simple linear MLM with one predictor, random intercepts, and random slopes can summarised with the following equation:

$$y_{ij} = \gamma_{00} + \gamma_{10}x_{ij} + U_{0j} + U_{1j} + R_{ij}, \tag{S1}$$

where y is some continuous outcome variable, x is a predictor variable, i is the observation, j is the cluster, γ_{00} is the overall ("biggie") intercept term, γ_{10} is a fixed effect for x on y, U_{0j} is a random deviation from γ_{00} for cluster j, U_{1j} is a random deviation from γ_{10} for cluster j, and R_{ij} is within-cluster residual.

This is all well and good, but isn't very concise. That is, we have to write many terms to represent even a simple model. As we see in the main text, it can be useful to represent coded categorical variables in a more concise way. To make things more compact, we can use **matrix notation** to represent the same equation:

$$\mathbf{Y} = \mathbf{XB} + \mathbf{Zu} + \mathbf{R}. \tag{S2}$$

Here, we have abstracted away the individual coefficients and information about level 1 and level 2 (the γ, i, and j from equation S2) into a series of vectors. $\mathbf{Y}$ is a vector of the y values, $\mathbf{X}$ is a design matrix for the fixed effects, $\mathbf{B}$ is a vector of fixed effects, γ, Z is a design matrix for the random effects, $\mathbf{u}$ is vector of random effects, U_{kj}, and $\mathbf{R}$ is a vector of the normally distributed residuals, R_{ij}. This is more clearly illustrated when we break apart the matrix as below. Imagine there are 2 observations (i) in 3 groups (j). The resultant matrix would look like this[7]:

[7] the extra column of 1s in $\mathbf{X}$ represents the intercept: $\gamma_{00} \times 1 = \gamma_{00}$

$$\underbrace{\begin{bmatrix} y_{11} \\ y_{12} \\ y_{21} \\ y_{22} \\ y_{31} \\ y_{32} \end{bmatrix}}_{\mathbf{Y}} = \underbrace{\begin{bmatrix} 1 & x_{11} \\ 1 & x_{12} \\ 1 & x_{21} \\ 1 & x_{22} \\ 1 & x_{31} \\ 1 & x_{32} \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{pmatrix} \gamma_{00} \\ \gamma_{10} \end{pmatrix}}_{\mathbf{B}} + \underbrace{\begin{bmatrix} 1 & x_{11} \\ 1 & x_{12} \\ 1 & x_{21} \\ 1 & x_{22} \\ 1 & x_{31} \\ 1 & x_{32} \end{bmatrix}}_{\mathbf{Z}} \underbrace{\begin{bmatrix} U_{01} & U_{11} \\ U_{01} & U_{11} \\ U_{02} & U_{12} \\ U_{02} & U_{12} \\ U_{03} & U_{13} \\ U_{03} & U_{13} \end{bmatrix}}_{\mathbf{u}} + \underbrace{\begin{bmatrix} R_{11} \\ R_{21} \\ R_{12} \\ R_{22} \\ R_{13} \\ R_{23} \end{bmatrix}}_{\mathbf{R}} \tag{S3}$$

To make the link between equation S4 and S2 explicit, if we evaluate both equations when $j = 2$, we get the same outcome:

$$\mathbf{Y_2} = \mathbf{X_2 B} + \mathbf{Z_2 u_2} + \mathbf{R_2}$$
$$=$$
$$y_{i2} = \gamma_{00} + \gamma_{10} x_{i2} + U_{02} + U_{12} + R_{i2}.$$

This may seem like many steps to arrive at basically the same conclusion, but keeping this compact notation in mind helps when we have to represent many categorical variables with many levels in equations. To do so, we would could summarise all the γ_{1x} with the simple $\mathbf{B}$ vector and all the coded categorical variables by the $\mathbf{X}$ and $\mathbf{Z}$ matrices. Furthermore, for pedagogical purposes, this matrix notation also shows up often in the literature, so it is helpful to know how it relates to the material we covered in class.

References

Goldstein, H., Browne, W., & Rasbash, J. (2002). Partitioning variation in generalised linear multilevel models. *Understanding Statistics, 1.* 223–232.

Goldstein, H. and Rasbash, J. (1996). Improved approximations for multilevel models with binary responses. *Journal of the Royal Statistical Society, A. 159.* 505-13.

Hilbe, J. M. (2013). Can binary logistic models be over-dispersed?. *Jet Propulsion Laboratory, California Institute of Technology and Arizona State University.* Retrieved from: http://highstat.com/Books/BGS/GLMGLMM/pdfs/HILBE-Can_binary_logistic_models_be_overdispersed2Jul2013.pdf

Inzlicht, M., Shenhav, A., & Olivola, C. Y. (2018). The effort paradox: Effort is both costly and valued. *Trends in cognitive sciences, 22*(4), 337-349.

Kool, W., McGuire, J. T., Rosen, Z. B., & Botvinick, M. M. (2010). Decision making and the avoidance of cognitive demand. *Journal of experimental psychology: general, 139*(4), 665.