Logistic Mixed Effects

But first, some LRM & GLM stuff
LRM vs. GLM

SampleSize = 1000

PollData = data.frame(Subj = 1:SampleSize,
   Sex = rbinom(SampleSize, 1, .55), # 55% as 1, 45% as 0
   Ed=sample(1:4, # 1 – <HS, 2-HS, 3-College, 4-Uni
      size=SampleSize,
      prob=c(.154, .239, .377, .229), # Probability of each category
      replace=T))

PollData = cbind(PollData, EdS=c("<HS", "HS", "C", "U")[PollData$Ed], Yes = rbinom
   (SampleSize, 1, prob = .5 + .05*PollData$Sex + c(-.2, -.3, .05, .3)[PollData$Ed]))

<table>
<thead>
<tr>
<th>Subj</th>
<th>Sex</th>
<th>Ed</th>
<th>EdS</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>U</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>&lt;HS</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>U</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>U</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>&lt;HS</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>&lt;HS</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>&lt;HS</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>0</td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>0</td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>1</td>
<td>2</td>
<td>HS</td>
</tr>
</tbody>
</table>
LRM vs. GLM

```r
> glm(Yes~EdS*Sex, data=PollData, family=binomial)
```

Coefficients:

|                  | Estimate | Std. Error | z value | Pr(>|z|) |
|------------------|----------|------------|---------|----------|
| (Intercept)      | -0.57536 | 0.24056    | -2.392  | 0.01677  * |
| EdSC             | 0.85162  | 0.29072    | 2.929   | 0.00340  ** |
| EdSHS            | -0.95482 | 0.34029    | -2.806  | 0.00502  ** |
| EdSU             | 2.23079  | 0.35767    | 6.237   | 4.46e-10 *** |
| Sex              | 0.14928  | 0.33095    | 0.451   | 0.65194  |
| EdSC:Sex         | -0.06056 | 0.39540    | -0.153  | 0.87827  |
| EdSHS:Sex        | 0.56415  | 0.45321    | 1.245   | 0.21321  |
| EdSU:Sex         | 0.15787  | 0.49666    | 0.318   | 0.75058  |

```r
> lrm(Yes~EdS*Sex, data=PollData)
```

<table>
<thead>
<tr>
<th></th>
<th>Coef</th>
<th>S.E.</th>
<th>Wald</th>
<th>Z</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>-0.57536</td>
<td>0.2406</td>
<td>-2.39</td>
<td>0.0168</td>
<td></td>
</tr>
<tr>
<td>EdS=C</td>
<td>0.85162</td>
<td>0.2907</td>
<td>2.93</td>
<td>0.0034</td>
<td></td>
</tr>
<tr>
<td>EdS=HS</td>
<td>-0.95482</td>
<td>0.3403</td>
<td>-2.81</td>
<td>0.0050</td>
<td></td>
</tr>
<tr>
<td>EdS=U</td>
<td>2.23079</td>
<td>0.3577</td>
<td>6.24</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>Sex</td>
<td>0.14928</td>
<td>0.3310</td>
<td>0.45</td>
<td>0.6519</td>
<td></td>
</tr>
<tr>
<td>EdS=C * Sex</td>
<td>-0.06056</td>
<td>0.3954</td>
<td>-0.15</td>
<td>0.8783</td>
<td></td>
</tr>
<tr>
<td>EdS=HS * Sex</td>
<td>0.56415</td>
<td>0.4532</td>
<td>1.24</td>
<td>0.2132</td>
<td></td>
</tr>
<tr>
<td>EdS=U * Sex</td>
<td>0.15787</td>
<td>0.4967</td>
<td>0.32</td>
<td>0.7506</td>
<td></td>
</tr>
</tbody>
</table>
LRM vs. GLM

• GLM & LRM are identical

• Wald Z – It turns out that when data aren’t normal there is a way to calculate a score that is approximated by the Z distribution \([N(0, 1), G(0, 1)]\)
  – Caveat! You should have at least 5 “Yes” and 5 “No” in each cell. Less than that, and the Wald Z is not very good.
Long vs. Wide

• GLM will take either long or wide data
  – formula = Accuracy ~ ...
  – formula = cbind(nCorrect, nIncorrect) ~ ...

• LRM can only handle long data

• (See LRMvGLM.R)
Logistic MEM

• Nearly identical to MEM (using lmer)

• In practice, the only differences are:
  – Inputs (Y values, or your DV) are 1s and 0s
    • Or you can use [Correct, Incorrect], if that’s appropriate to your data
  – Requires `family="binomial"`, just like logistic regression
Baayen’s English Dative Data

• Analysis of natural language, and how the dative case is used
  – Mary gave John the book. (NP)
  – Mary gave the book to John. (PP)

• Coded each use of the dative for
  – Whether it was NP or PP
  – Which verb (give), and the speaker (most of the time)
  – A whole tonne of other things.
### Dative Data

```r
library(languageR)  # gives us the 'dative' dataset
dative = dative[dative$Modality==‘spoken’,]  # consider only the spoken data
head(dative)
```

<table>
<thead>
<tr>
<th>Speaker</th>
<th>Modality</th>
<th>Verb</th>
<th>SemanticClass</th>
<th>LengthOfRecipient</th>
<th>AnimacyOfRec</th>
<th>DefinOfRec</th>
</tr>
</thead>
<tbody>
<tr>
<td>903</td>
<td>S1176</td>
<td>spoken give</td>
<td>a</td>
<td>1</td>
<td>inanimate</td>
<td>definite</td>
</tr>
<tr>
<td>904</td>
<td>S1110</td>
<td>spoken give</td>
<td>c</td>
<td>2</td>
<td>animate</td>
<td>indefinite</td>
</tr>
<tr>
<td>905</td>
<td>S1110</td>
<td>spoken pay</td>
<td>a</td>
<td>1</td>
<td>animate</td>
<td>definite</td>
</tr>
<tr>
<td>906</td>
<td>S1146</td>
<td>spoken give</td>
<td>a</td>
<td>2</td>
<td>animate</td>
<td>definite</td>
</tr>
<tr>
<td>907</td>
<td>S1146</td>
<td>spoken give</td>
<td>t</td>
<td>2</td>
<td>animate</td>
<td>definite</td>
</tr>
<tr>
<td>908</td>
<td>S1053</td>
<td>spoken give</td>
<td>t</td>
<td>1</td>
<td>animate</td>
<td>definite</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PronomOfRec</th>
<th>LengthOfTheme</th>
<th>AnimacyOfTheme</th>
<th>DefinOfTheme</th>
<th>PronomOfTheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>903</td>
<td>pronominal</td>
<td>2</td>
<td>inanimate</td>
<td>indefinite</td>
</tr>
<tr>
<td>904</td>
<td>nonpronominal</td>
<td>1</td>
<td>inanimate</td>
<td>definite</td>
</tr>
<tr>
<td>905</td>
<td>pronominal</td>
<td>1</td>
<td>inanimate</td>
<td>indefinite</td>
</tr>
<tr>
<td>906</td>
<td>nonpronominal</td>
<td>4</td>
<td>inanimate</td>
<td>indefinite</td>
</tr>
<tr>
<td>907</td>
<td>nonpronominal</td>
<td>3</td>
<td>inanimate</td>
<td>definite</td>
</tr>
<tr>
<td>908</td>
<td>pronominal</td>
<td>2</td>
<td>inanimate</td>
<td>definite</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RealizationOfRecipient</th>
<th>AccessOfRec</th>
<th>AccessOfTheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>903</td>
<td>NP</td>
<td>given</td>
</tr>
<tr>
<td>904</td>
<td>PP</td>
<td>new</td>
</tr>
<tr>
<td>905</td>
<td>PP</td>
<td>given</td>
</tr>
<tr>
<td>906</td>
<td>NP</td>
<td>accessible</td>
</tr>
<tr>
<td>907</td>
<td>PP</td>
<td>accessible</td>
</tr>
<tr>
<td>908</td>
<td>PP</td>
<td>given</td>
</tr>
</tbody>
</table>
Dative Data

• Which variables predict whether the Speaker will use NP or PP?
  – RealizationOfRecipient

• Since this is a binary DV, we can use logistic regression
LRM

```r
> dative.lr = glm(RealizationOfRecipient ~ AccessOfTheme + AccessOfRec + LengthOfRecipient + AnimacyOfRec + AnimacyOfTheme + PronomOfTheme + DefinOfTheme + LengthOfTheme + SemanticClass + Modality, data=dative, family="binomial")
> summary(dative.lr)
```

Coefficients:  

|                           | Estimate | Std. Error | z value | Pr(>|z|)  |
|---------------------------|----------|------------|---------|-----------|
| (Intercept)               | 0.3992   | 0.5645     | 0.707   | 0.47945   |
| AccessOfThemegiven        | 1.1429   | 0.2586     | 4.420   | 9.88e-06  *** |
| AccessOfThemenew          | -0.2642  | 0.2381     | -1.110  | 0.26720   |
| AccessOfRecgiven          | -2.3832  | 0.2124     | -11.222 | < 2e-16   *** |
| AccessOfRecnew            | -0.2731  | 0.3326     | -0.821  | 0.41148   |
| LengthOfRecipient         | 0.5442   | 0.0891     | 6.112   | 9.86e-10  *** |
| AnimacyOfRecinanimate     | 2.7649   | 0.2912     | 9.494   | 9.86e-10  *** |
| AnimacyOfThemeinanimate   | -1.1629  | 0.4486     | -2.592  | 0.00954   ** |
| PronomOfThemepronominial  | 1.7745   | 0.2304     | 7.701   | 1.35e-14  *** |
| DefinOfThemeindefinite    | -0.9972  | 0.2147     | -4.645  | 3.41e-06  *** |
| LengthOfTheme             | -0.1779  | 0.0360     | -4.945  | 7.63e-07  *** |
| SemanticClassc            | -1.0429  | 0.3073     | -3.394  | 0.00069   *** |
| SemanticClassf            | 0.5582   | 0.4789     | 1.166   | 0.24374   |
| SemanticClassp            | -2.9151  | 1.1369     | -2.564  | 0.01035   * |
| SemanticClasss            | 1.2599   | 0.2088     | 6.034   | 1.60e-09  *** |

• Everything is significant
LMER (Logistic Mixed Effects)

```
> dative.lmer.s = lmer(RealizationOfRecipient ~ AccessOfTheme +
AccessOfRec + LengthOfRecipient + AnimacyOfRec + AnimacyOfTheme +
PronomOfTheme + DefinOfTheme + LengthOfTheme + SemanticClass +
(1|Speaker), data = dative, family = "binomial")
```

**Random effects:**
- Groups: Speaker
- Name: (Intercept)
- Variance: 4.9157e-11
- Std.Dev.: 7.0112e-06

**Number of obs:** 2360, groups: Speaker, 424

**Fixed effects:**

| Estimate | Std. Error | z value | Pr(> |z|) |
|----------|------------|---------|-------|
| (Intercept) | 0.39918 | 0.56447 | 0.707 | 0.47946 |
| AccessOfTheme Given | 1.14297 | 0.25861 | 4.420 | 9.88e-06 *** |
| AccessOfThem New | -0.26418 | 0.23811 | -1.110 | 0.26721 |
| AccessOfRec Given | -2.38317 | 0.21237 | -11.222 | < 2e-16 *** |
| AccessOfRec New | -0.27311 | 0.33255 | -0.821 | 0.41149 |
| LengthOfRecipient | 0.54423 | 0.08905 | 6.112 | 9.86e-10 *** |
| AnimacyOfRec Inanimate | 2.76495 | 0.29123 | 9.494 | < 2e-16 *** |
| AnimacyOfTheme Inanimate | -1.16288 | 0.44862 | -2.592 | 0.00954 ** |
| PronomOfTheme Pronominal | 1.77449 | 0.23044 | 7.701 | 1.36e-14 *** |
| DefinOfTheme Indefinite | -0.99717 | 0.21469 | -4.645 | 3.41e-06 *** |
| LengthOfTheme | -0.17798 | 0.03600 | -4.945 | 7.63e-07 *** |
| SemanticClass c | -1.04294 | 0.30732 | -3.394 | 0.00069 *** |
| SemanticClass f | 0.55824 | 0.47889 | 1.166 | 0.24374 |
| SemanticClass p | -2.91509 | 1.13693 | -2.564 | 0.01035 * |
| SemanticClass t | 1.25987 | 0.20881 | 6.034 | 1.60e-09 *** |

How much is just Speaker differences?
How much does the verb predict the realization?

```r
> dative.lmer.v = lmer(RealizationOfRecipient ~ AccessOfTheme + AccessOfRec + LengthOfRecipient + AnimacyOfRec + AnimacyOfTheme + PronomOfTheme + DefinOfTheme + LengthOfTheme + SemanticClass + (1 | Verb), data = dative, family = "binomial")

Random effects:

Groups  Name        Variance  Std.Dev.
  Verb   (Intercept)  4.3753  2.0917

Number of obs: 2360, groups: Verb, 38

Fixed effects:

                     Estimate Std. Error  z value  Pr(>|z|)
(Intercept)       0.58596    0.78446   0.747   0.4551
AccessOfThemegiven 1.56012    0.28816   5.414  6.16e-08 ***
AccessOfThemew new -0.30057    0.27429  -1.096   0.2732
AccessOfRecgiven   -2.61477    0.24063  -10.866 < 2e-16 ***
AccessOfRecnew     -0.54845    0.37397  -1.467   0.1425
LengthOfRecipient   0.59265    0.09433   6.283   3.32e-10 ***
AnimacyOfRecanimate 2.52902    0.34427   7.346  2.04e-13 ***
AnimacyOfThemeanimate -1.16563    0.52323  -2.228    0.0259 *
PronomOfThemepronominal 2.47834    0.26185   9.465  < 2e-16 ***
DefinOfTheme indefinite -1.05884    0.30431  -4.406    1.05e-05 ***
LengthOfTheme      -0.18548    0.03972  -4.670    3.01e-06 ***
SemanticClassc     0.34392    0.45427   0.757    0.4490
SemanticClassf     0.36428    0.84381   0.432    0.6659
SemanticClassp     -4.01978    2.15247  -1.868    0.0618 .
SemanticClasst     0.19525    0.26717   0.731    0.4649
```
LMER (Logistic Mixed Effects)

OK, so now we can test whether or not Semantic Class is still contributing anything.

```r
>dative.lmer.v = lmer(RealizationOfRecipient ~ AccessOfTheme + AccessOfRec + LengthOfRecipient + AnimacyOfRec + AnimacyOfTheme + PronomOfTheme + DefinOfTheme + LengthOfTheme + SemanticClass + (1|Verb), data=dative, family="binomial")

>dative.lmer.v2 = lmer(RealizationOfRecipient ~ AccessOfTheme + AccessOfRec + LengthOfRecipient + AnimacyOfRec + AnimacyOfTheme + PronomOfTheme + DefinOfTheme + LengthOfTheme + (1|Verb), data=dative, family="binomial")

>anova(dative.lmer.v, dative.lmer.v2)

Data: dative
Models:
  dative.lmer.v2: RealizationOfRecipient ~ AccessOfTheme + AccessOfRec + LengthOfRecipient + ... + (1 | Verb)
  dative.lmer.v: RealizationOfRecipient ~ AccessOfTheme + AccessOfRec + LengthOfRecipient + ... + SemanticClass + (1 | Verb)

             Df  AIC    BIC   logLik Chisq Chi Df Pr(>Chisq)
 dative.lmer.v2 12 1012.0 1081.2  -493.99
 dative.lmer.v  16 1013.8 1106.1  -490.91  6.1586      4     0.1876
```
A few other issues

• A few issues that I’ve run into:
  – How to decide if your mixed effects model is outperforming the regression model (applies to linear and logistic models)
  – How to test for a single, simple effect (same)
  – Interpreting and graphing logistic coefficients
LMER vs. LRM/GLM

I recommend using the AIC, and choosing the smaller one.

```r
>dative.lr = glm(RealizationOfRecipient ~ AccessOfTheme + AccessOfRec + LengthOfRecipient + AnimacyOfRec + AnimacyOfTheme + PronomOfTheme + DefinOfTheme + LengthOfTheme + SemanticClass, 
data=dative, family="binomial")
>dative.lmer.v = lmer(RealizationOfRecipient ~ AccessOfTheme + AccessOfRec + LengthOfRecipient + AnimacyOfRec + AnimacyOfTheme + PronomOfTheme + DefinOfTheme + LengthOfTheme + SemanticClass + (1|Verb), data=dative, family="binomial")

>AIC(dative.lr)
[1] 1149.301
>AIC(dative.lmer.v)
[1] 1013.816

># The dative.lmer.v model is clearly producing a smaller AIC

># Compare with the AIC for the "Speaker" model
>AIC(dative.lmer.s)
[1] 1151.301
```
Reading Accuracy Data

• 66 Subjects see 120 words
  – 60 have a high frequency neighbour (SAFE – *same*)
  – 60 do not have a high frequency neighbour (MILK)
  – (Not my data)

• Subjects are children, and the DV is whether or not they read the word aloud correctly.
Reading Accuracy Data

```r
HFN = read.csv("NeighbourFreq.csv") # read the dataset into HFN

# First, need to do some recoding
HFN$Acc = c(1,1,0,0,NA)[as.integer(HFN$RESP)] # 1 - Correct, 0 - Error, NA - misfire
HFN$HFNeigh = as.numeric(HFN$type!=0) # 1 - has a high frequency neighbour, 0 - does not
HFN$ppnr = factor(HFN$ppnr)
HFN$PrevAcc[2:length(HFN$Acc)]=HFN$Acc[1:(length(HFN$Acc)-1)]; HFN$PrevAcc[HFN$order==1]=NA
HFN$order.c = HFN$order – mean(HFN$order) # Centered version of the ‘order’ variable

# Drop the "misfire" data
HFN = HFN[!is.na(HFN$Acc),]
```

<table>
<thead>
<tr>
<th>ppnr</th>
<th>triallist</th>
<th>RT</th>
<th>type</th>
<th>item</th>
<th>RESP</th>
<th>order</th>
<th>Acc</th>
<th>HFNeigh</th>
<th>PrevAcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
<td>13</td>
<td>4</td>
<td>loud</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>NA</td>
</tr>
<tr>
<td>2</td>
<td>101</td>
<td>108</td>
<td>0</td>
<td>grow</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>101</td>
<td>95</td>
<td>0</td>
<td>news</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>101</td>
<td>26</td>
<td>3</td>
<td>stew</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>11</td>
<td>4</td>
<td>pink</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>101</td>
<td>93</td>
<td>0</td>
<td>real</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
# LMER vs. LRM/GLM

Looks like we should be using the LMER model

Also, PrevAcc seems not to be important (but let’s check)

```r
> HFN.lr = glm(Acc~HFNeigh+PrevAcc, data=HFN, family="binomial")
> HFN.lmer.s = lmer(Acc~HFNeigh+PrevAcc+(1|ppnr), data=HFN, family="binomial")
> AIC(HFN.lr)
[1] 4838.476
> AIC(HFN.lmer.s)
[1] 4247.450
> # The HFN.lmer.s model is clearly producing a smaller AIC

> print(HFN.lmer.s, corr=F)
Generalized linear mixed model fit by the Laplace approximation
Formula: Acc ~ HFNeigh + PrevAcc + (1 | ppnr)
   Data: HFN
     AIC   BIC logLik deviance
4247 4275  -2120     4239
Random effects:
   Groups   Name        Variance Std.Dev.
          ppnr (Intercept) 1.6259   1.2751
Number of obs: 7717, groups: ppnr, 66

Fixed effects:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        2.80455    0.19902  14.092  < 2e-16 ***
HFNeigh           -0.24177    0.08264  -2.925   0.00344 **
PrevAcc           -0.09306    0.11063   -0.841    0.40024
```
Why We Do Model Testing

Fit models with, and without PrevAcc

Compare with anova()

> HFN.lmer.s = lmer(Acc~HFNeigh+\texttt{PrevAcc}+(1|ppnr), data=HFN, family="binomial")
> HFN.lmer1.s = lmer(Acc~HFNeigh+(1|ppnr), data=HFN, family="binomial")

> anova(HFN.lmer.s, HFN.lmer1.s)

Data: HFN
Models:
HFN.lmer1.s: Acc ~ HFNeigh + (1 | ppnr)
HFN.lmer.s: Acc ~ HFNeigh + PrevAcc + (1 | ppnr)

<table>
<thead>
<tr>
<th>Df</th>
<th>AIC</th>
<th>BIC</th>
<th>logLik</th>
<th>Chisq</th>
<th>Chi Df</th>
<th>Pr(&gt;Chisq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HFN.lmer1.s</td>
<td>3</td>
<td>4378.5</td>
<td>4399.4</td>
<td>-2186.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HFN.lmer.s</td>
<td>4</td>
<td>4247.4</td>
<td>4275.3</td>
<td>-2119.7</td>
<td>133.03</td>
<td>1 &lt; 2.2e-16</td>
</tr>
</tbody>
</table>

# Clearly PrevAcc does matter.

# This data has lots of cells with few entries (few errors). As a result, Wald Z can’t be trusted. Need to use model testing.
Testing for a Simple Effect

• Let’s assume that only HFNeigh matters

• How to test for simple effect of just HFNeigh?

• Can’t rely on Wald Z score

```r
# To test whether or not HFNeigh matters, create a "dumb" model that assumes only an intercept. Test whether or not HFNeigh is better than just the average.

> HFN.lmer.s = lmer(Acc ~ HFNeigh + (1|ppnr), data=HFN, family="binomial")
> HFN.dumb = lmer(Acc ~ 1 + (1|ppnr), data=HFN, family="binomial")  # only the intercept is fit

> anova(HFN.dumb, HFN.lmer.s)

Data: HFN
Models:
  HFN.dumb: Acc ~ 1 + (1 | ppnr)
  HFN.lmer.s: Acc ~ HFNeigh + (1 | ppnr)

Df    AIC    BIC  logLik  Chisq Chi Df Pr(>Chisq)
HFN.dumb    2 4384.3 4398.3 -2190.2
HFN.lmer.s  3 4378.5 4399.4 -2186.2 7.8408      1   0.005108
```
Coefficients & Graphing

> print(HFN.lmer.s, corr=F)

Generalized linear mixed model
Formula: Acc ~ HFNeigh + PrevAcc + (1 | ppnr)
   Data: HFN
   AIC  BIC logLik deviance
4247 4275   -2120     4239
Random effects:
Groups     Name        Variance Std.Dev.
ppnr   (Intercept) 1.6259   1.2751
Number of obs: 7717, groups: ppnr, 66

Fixed effects:
     Estimate Std. Error  z value Pr(>|z|)
(Intercept)  2.80455   0.19902  14.092  < 2e-16
HFNeigh      -0.24177   0.08264  -2.925   0.00344
PrevAcc       0.09306   0.11063   0.841   0.40024

• Let’s assume the model to the left is the best.

• The coefficients have no obvious meaning (other than the direction)

• How do you recover the “probabilities”?

• This does involve a little bit of algebra (sorry)
We want something like this!

Table 1: Proportion accuracy, by condition

<table>
<thead>
<tr>
<th></th>
<th>Correct Previous Trial</th>
<th>Incorrect Previous Trial</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Frequency Neighbour</td>
<td>0.956</td>
<td>0.897</td>
</tr>
<tr>
<td>No HF Neighbour</td>
<td>0.965</td>
<td>0.917</td>
</tr>
</tbody>
</table>

Figure 1: Percentage correct, by condition
What Did We Fit?

\[ Y = \beta_0 + \beta_1 \times HFNeigh + \beta_2 \times PrevAcc + (\mu_{subject}) + \varepsilon \]

- $\beta_0$ is just the intercept, or $2.8$
- $\beta_1$ is the effect of HFNeigh, or $-0.24$
- $\beta_2$ is the effect of PrevAcc, or $0.093$

- Ignore the rest ($\mu_{subject}$ is the ‘random effect’)
That’s most of what we need

\[ Y = 2.4 - 0.24 \times HFNeigh + 0.093 \times PrevAcc \]

- We can rewrite the equation, using real numbers.
- Remember, HFNeigh is 0 (No neighbour) or 1 (a high frequency neighbour)
- PrevAcc is also 0 (Error), or 1 (Correct)
- It’s not that hard to get all of the « L » values now.
That’s most of what we need

\[ Y = 2.4 - 0.24 \times HFNeigh + 0.093 \times PrevAcc \]

• say we want HFNeigh = 1 (has a HF neighbour), and PrevAcc = 0 (got the previous word wrong)

\[ Y = 2.4 - 0.24 \times 1 + 0.093 \times 0 = 2.16 \]

• We can do the same for all combinations of HFNeigh and PrevAcc
We get this

<table>
<thead>
<tr>
<th></th>
<th>Correct Previous Trial</th>
<th>Incorrect Previous Trial</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Frequency Neighbour</td>
<td>3.09</td>
<td>2.16</td>
</tr>
<tr>
<td>No HF Neighbour</td>
<td>3.33</td>
<td>2.40</td>
</tr>
</tbody>
</table>

- If this is a LINEAR model, then we are done.
We get this

<table>
<thead>
<tr>
<th></th>
<th>Correct Previous Trial</th>
<th>Incorrect Previous Trial</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Frequency</td>
<td>3.09</td>
<td>2.16</td>
</tr>
<tr>
<td>Neighbour</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No HF Neighbour</td>
<td>3.33</td>
<td>2.40</td>
</tr>
</tbody>
</table>

• We have to go back to the link function

\[ Y = \log\left(\frac{p}{1 - p}\right) = 2.16 \]

• Turns out we can easily get \( p \) out of that.

\[ p = \frac{1}{1 + e^{-2.16}} = 0.897 \]

• If it’s a logistic model, we have one more step to get \( p \).
Those are the meaningful values

Table 1: Accuracy, by condition

<table>
<thead>
<tr>
<th></th>
<th>Correct Previous Trial</th>
<th>Incorrect Previous Trial</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Frequency Neighbour</td>
<td>0.956</td>
<td>0.897</td>
</tr>
<tr>
<td>No HF Neighbour</td>
<td>0.965</td>
<td>0.917</td>
</tr>
</tbody>
</table>

Figure 1: Percentage correct, by condition
Getting proportions back in R

> print(HFN.lmer.s, corr=F)

Generalized linear mixed model
Formula: Acc ~ HFNeigh + PrevAcc + (1 | ppnr)
  Data: HFN
  AIC  BIC logLik deviance
  4247 4275  -2120     4239
Random effects:
  Groups Name        Variance Std.Dev.
    ppnr   (Intercept) 1.6259   1.2751
Number of obs: 7717, groups: ppnr, 66

Fixed effects:
  Estimate Std. Error z value  Pr(>|z|)
(Intercept)  2.80455   0.19902  14.092  < 2e-16
HFNeigh     -0.24177   0.08264  -2.925  0.00344
PrevAcc     0.09306   0.11063   0.841   0.40024

• Using the values from the table, we could calculate the various values we want to plot.
• From that we need one more step:

> z = c(3.09, 2.16, 3.33, 2.4)
> p = 1/(1+exp(-z))
> p
[1] 0.956 0.897 0.965 0.917