Lecture 5: Beyond Linear Models

LSA 2013, LI539

Mixed Effect Models

T. Florian Jaeger

Brain and Cognitive Sciences
University of Rochester

July 16, 2013

1. **Logit Models**
    - Example
    - Logistic Regression
    - Mixed
2. **Quasi-binomial**
    - Ordinary
3. **Poisson models**
    - Ordinary
    - Mixed
4. **Quasi-Poisson**
    - Ordinary
5. **Multivariate models**
    - Multivariate Gaussians
6. **Multinomial models**
    - Ordered data

Reviewing Logistic Regression

Recall that **logistic regression** is a kind of **GLM** (with a binomial link function).

- The linear predictor:

$$\eta = \alpha + \beta_1 \mathbf{x_1} + \cdots + \beta_n \mathbf{x_n}$$

Reviewing Logistic Regression

Recall that **logistic regression** is a kind of **GLM** (with a binomial link function).

- The linear predictor:

$$\eta = \alpha + \beta_1 \mathbf{x_1} + \cdots + \beta_n \mathbf{x_n}$$

- The link function g is the logit transform:

$$\mathrm{E}(\mathbf{y}) = p = \mathrm{g}^{-1}(\eta) \Leftrightarrow$$
$$\mathrm{g}(p) = \ln \frac{p}{1-p} = \eta = \alpha + \beta_1 \mathbf{x_1} + \cdots + \beta_n \mathbf{x_n} \quad (1)$$

Reviewing Logistic Regression

Recall that **logistic regression** is a kind of **GLM** (with a binomial link function).

- The linear predictor:

$$\eta = \alpha + \beta_1 \mathbf{x_1} + \cdots + \beta_n \mathbf{x_n}$$

- The link function g is the logit transform:

$$\begin{aligned} \mathrm{E}(\mathbf{y}) = p = \mathrm{g}^{-1}(\eta) \Leftrightarrow \\ \mathrm{g}(p) = \ln \frac{p}{1-p} = \eta = \alpha + \beta_1 \mathbf{x_1} + \cdots + \beta_n \mathbf{x_n} \end{aligned} \quad (1)$$

- The distribution around the mean is taken to be binomial.

Alternative descriptions

- If our linear predictor describes expected log-odds . . .

$$\eta = \ln \frac{p}{1-p} = \mathbf{X}\beta \tag{2}$$

- . . . then . . .

Alternative descriptions

- If our linear predictor describes expected log-odds ...

$$\eta = \ln \frac{p}{1-p} = \mathbf{X}\beta \quad (2)$$

- ... then ...

$$\begin{aligned}
& \tfrac{p}{1-p} && = \exp^{\mathbf{X}\beta} && (3)\\
\Leftrightarrow\quad & p && = \exp^{\mathbf{X}\beta} - p\exp^{\mathbf{X}\beta} && (4)\\
\Leftrightarrow\quad & p + p\exp^{\mathbf{X}\beta} && = \exp^{\mathbf{X}\beta} && (5)\\
\Leftrightarrow\quad & p(1+\exp^{\mathbf{X}\beta}) && = \exp^{\mathbf{X}\beta} && (6)
\end{aligned}$$

Alternative descriptions

- If our linear predictor describes expected log-odds ...

$$\eta = \ln \frac{p}{1-p} = \mathbf{X}\beta \quad (2)$$

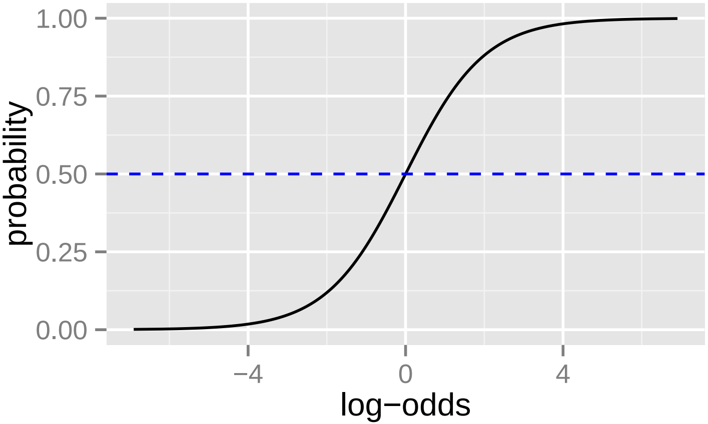
- ... then ...

$$\begin{aligned} & \tfrac{p}{1-p} && = \exp^{\mathbf{X}\beta} && (3) \\ \Leftrightarrow \quad & p && = \exp^{\mathbf{X}\beta} - p\exp^{\mathbf{X}\beta} && (4) \\ \Leftrightarrow \quad & p + p\exp^{\mathbf{X}\beta} && = \exp^{\mathbf{X}\beta} && (5) \\ \Leftrightarrow \quad & p(1+\exp^{\mathbf{X}\beta}) && = \exp^{\mathbf{X}\beta} && (6) \end{aligned}$$

- which brings us to two forms you sometimes see:

$$\begin{aligned} p &= \frac{\exp^{\mathbf{X}\beta}}{\mathbf{1}+\exp^{\mathbf{X}\beta}} \\ &= \frac{1}{1+\frac{1}{\exp^{\mathbf{X}\beta}}} \\ &= \frac{\mathbf{1}}{\mathbf{1}+\exp^{-\mathbf{X}\beta}} \end{aligned}$$

Visualization of the relation between log-odds and probabilities



- This relation is particularly clear in the following form of the model:

$$p = \frac{1}{1 + \exp^{-\mathbf{X}\beta}} \quad (7)$$

Data 2: Lexical decision *response*

- **Outcome:** Correct or incorrect response (`Correct`)
- **Inputs:** same as in linear model

```
library(languageR)
data(lexdec)
# to learn more about this data, use ?lexdec
# randomly sampling rows from lexdec
head(lexdec[sample(x=1:nrow(lexdec), size=20, replace=F),c(1:6, 9:10)])

##      Subject    RT Trial Sex NativeLanguage Correct     Word Frequency
## 1058       S 6.385    87   F        English correct   banana     5.024
## 654       W1 6.290    76   M        English correct    apple     6.304
## 52        A1 6.435   133   F        English correct     leek     3.332
## 1439       V 6.603    57   F          Other correct     leek     3.332
## 190        P 6.423    87   F          Other correct   donkey     5.541
## 1646      M2 6.323   161   F          Other correct mushroom     5.537
```

Preliminaries

- Recall that R by default interprets factors levels in alphanumeric order (cf. Maureen Gillespie's lecture on **Coding**). This can also affect dependent variables and hence the interpretation of our model.
- This makes the default coding for `Correct` 0 for 'correct' and 1 for 'incorrect', which is confusing (the coefficient will encode the difference of incorrect, compared to correct, trials, rather than the other way around)
- So, let's reorder the levels:

```
lexdec$Correct = factor(lexdec$Correct, levels=c("incorrect", "correct"))
```

Logistic regression over individual cases

- Let's investigate the effect of native speaker status on the accuracy in lexical decision tasks.
- We again start with a simple model (notice that the noise term now is implicit in the link function, cf. the **linear model**):

$$\ln \frac{\mathbf{p}}{\mathbf{1-p}} = \overbrace{\beta_0 + \beta_1 \mathbf{NativeLanguage}}^{\text{Predicted Mean}}$$

Logistic regression over individual cases

$$\ln \frac{p}{1-p} = \overbrace{\beta_0 + \beta_1 \mathbf{NativeLanguage}}^{\text{Predicted Mean}}$$

```
g1 = glm(Correct ~ 1 + NativeLanguage, data=lexdec, family=binomial)
summary(g1)

##
## Call:
## glm(formula = Correct ~ 1 + NativeLanguage, family = binomial,
##     data = lexdec)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.654   0.245   0.245   0.327   0.327
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)             3.492      0.192   18.20   <2e-16 ***
## NativeLanguageOther   -0.590      0.256   -2.31    0.021 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 548.57  on 1658  degrees of freedom
## Residual deviance: 543.17  on 1657  degrees of freedom
## AIC: 547.2
##
## Number of Fisher Scoring iterations: 6
```

Proportion data

- Imagine, that the data was in a different format, giving us proportions (and number of total cases):

```
library(plyr)
lexdec2= ddply(lexdec,
        .(NativeLanguage),
        function(df) data.frame(
          Correct.mean=mean(ifelse(df$Correct=='correct', 1, 0)),
          Correct.count=length(df$Correct)
        )
)

lexdec2

##   NativeLanguage Correct.mean Correct.count
## 1        English       0.9705           948
## 2          Other       0.9480           711
```

Weighted logistic regression over proportions


```
g2 = glm(Correct.mean ~ 1 + NativeLanguage, data=lexdec2,
         family=binomial, weight=Correct.count)
summary(g2)

##
## Call:
## glm(formula = Correct.mean ~ 1 + NativeLanguage, family = binomial,
##     data = lexdec2, weights = Correct.count)
##
## Deviance Residuals:
## [1]  0  0
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)             3.492      0.192   18.20   <2e-16 ***
## NativeLanguageOther    -0.590      0.256   -2.31    0.021 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5.3939  on 1  degrees of freedom
## Residual deviance: 0.0000  on 0  degrees of freedom
## AIC: 14.55
##
## Number of Fisher Scoring iterations: 3
```

Count data

- Alternatively, we might have the the data in form of counts for ‘correct’ and ‘incorrect’:

```
library(plyr)
lexdec3= ddply(lexdec,
        .(NativeLanguage),
        function(df) data.frame(
          Correct.count=table(df$Correct)['correct'],
          Incorrect.count=table(df$Correct)['incorrect']
        )
)

lexdec3

##   NativeLanguage Correct.count Incorrect.count
## 1        English           920              28
## 2          Other           674              37
```

Logistic regression over counts


```
g3 = glm(cbind(Correct.count, Incorrect.count) ~
         1 + NativeLanguage, data=lexdec3,
         family=binomial)
summary(g3)

##
## Call:
## glm(formula = cbind(Correct.count, Incorrect.count) ~ 1 + NativeLanguage,
##     family = binomial, data = lexdec3)
##
## Deviance Residuals:
## [1]  0  0
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)            3.492      0.192   18.20   <2e-16 ***
## NativeLanguageOther   -0.590      0.256   -2.31    0.021 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5.3939  on 1  degrees of freedom
## Residual deviance: 0.0000  on 0  degrees of freedom
## AIC: 14.55
##
## Number of Fisher Scoring iterations: 3
```

Comparison

- Notice that all these models give us the same coefficient estimates, standard error estimates for those coefficients, and hence same significances.
- However, the models differ in their data likelihood – they are fit over different data.

Table: Comparison of different ways to fit logit model with binomial link function

	Dependent variable:		
	Correct	Correct.mean	cbind(Correct.count, Incorrect.count)
	(1)	(2)	(3)
Constant	3.492^{***}	3.492^{***}	3.492^{***}
	(0.192)	(0.192)	(0.192)
NativeLanguageOther	-0.590^{**}	-0.590^{**}	-0.590^{**}
	(0.256)	(0.256)	(0.256)
Observations	1,659	2	2
Log likelihood	−271.600	−5.273	−5.273
Akaike Inf. Crit.	547.200	14.550	14.550
Note:			*p<0.1; **p<0.05; ***p<0.01

Alternative functions for logistic regression

- Many R packages provide ways of fitting logistic regression.
  - nnet provides neural network tools that, among many other things, can be used to run logistic regression.
  - rms (formerly known as Design) is a package by Frank Harrell that provides a large number of convenience functions for a variety of GLMs, including logistic regresison. Check out lrm for a great logistic regression function.

```
library(rms)

## Warning:  package 'Hmisc' was built under R version 3.0.1

lrm(Correct ~ 1 + NativeLanguage, data=lexdec, x=T, y=T)

##
## Logistic Regression Model
##
## lrm(formula = Correct ~ 1 + NativeLanguage, data = lexdec, x = T,
##     y = T)
##
##                        Model Likelihood     Discrimination    Rank Discrim.
##                           Ratio Test           Indexes           Indexes
## Obs          1659    LR chi2      5.39    R2         0.012    C       0.573
##  incorrect     65    d.f.            1    g          0.289    Dxy     0.146
##  correct     1594    Pr(> chi2) 0.0202    gr         1.335    gamma   0.287
## max |deriv| 2e-12                         gp         0.011    tau-a   0.011
##                                           Brier      0.038
##
##                       Coef    S.E.   Wald Z Pr(>|Z|)
## Intercept              3.4922 0.1918 18.20  <0.0001
## NativeLanguage=Other -0.5899 0.2556 -2.31   0.0210
```

lrm


```
dd = datadist(lexdec)
options(datadist='dd')
l5 = lrm(Correct ~ 1 + Frequency, data=lexdec, x=T, y=T)
summary(l5)

##              Effects              Response : Correct
##
##  Factor       Low   High  Diff. Effect S.E. Lower 0.95 Upper 0.95
##  Frequency    3.951 5.652 1.701 0.94   0.18 0.59       1.29
##   Odds Ratio 3.951 5.652 1.701 2.56     NA 1.80       3.63

validate(l5, B=50)

##           index.orig training    test optimism index.corrected  n
## Dxy           0.3679   0.3720  0.3679   0.0041          0.3637 50
## R2            0.0625   0.0667  0.0625   0.0042          0.0582 50
## Intercept     0.0000   0.0000 -0.0930   0.0930         -0.0930 50
## Slope         1.0000   1.0000  1.0283  -0.0283          1.0283 50
## Emax          0.0000   0.0000  0.0248   0.0248          0.0248 50
## D             0.0171   0.0184  0.0171   0.0013          0.0158 50
## U            -0.0012  -0.0012  0.0003  -0.0016          0.0003 50
## Q             0.0183   0.0196  0.0168   0.0029          0.0155 50
## B             0.0368   0.0370  0.0368   0.0002          0.0366 50
## g             0.7990   0.8155  0.7990   0.0165          0.7826 50
## gp            0.0286   0.0291  0.0286   0.0005          0.0281 50
```

lrm cont'd)


```
dd = datadist(lexdec)
options(datadist='dd')
l6 = lrm(Correct ~ 1 + NativeLanguage*Frequency, data=lexdec, x=T, y=T)
validate(l6, B=10, bw=TRUE)

##
##  Backwards Step-down - Original Model
##
##  Deleted                    Chi-Sq d.f. P      Residual d.f. P      AIC
##  NativeLanguage * Frequency 1.52   1    0.2175 1.52     1    0.2175 -0.48
##
## Approximate Estimates after Deleting Factors
##
##                         Coef   S.E. Wald Z         P
## Intercept             1.0911 0.4779  2.283 2.242e-02
## NativeLanguage=Other -0.5975 0.2603 -2.296 2.170e-02
## Frequency             0.5505 0.1058  5.202 1.970e-07
##
## Factors in Final Model
##
## [1] NativeLanguage Frequency
##           index.orig training   test optimism index.corrected  n
## Dxy           0.4027   0.4496 0.3780   0.0716          0.3311 10
## R2            0.0740   0.1006 0.0686   0.0320          0.0421 10
## Intercept     0.0000   0.0000 0.4006  -0.4006          0.4006 10
## Slope         1.0000   1.0000 0.8472   0.1528          0.8472 10
## Emax          0.0000   0.0000 0.1215   0.1215          0.1215 10
## D             0.0205   0.0275 0.0189   0.0086          0.0119 10
## U            -0.0012  -0.0012 0.0009  -0.0021          0.0009 10
## Q             0.0217   0.0287 0.0180   0.0107          0.0110 10
## B             0.0365   0.0347 0.0367  -0.0020          0.0385 10
## g             0.8708   0.9331 0.7553   0.1778          0.6930 10
## gp            0.0309   0.0330 0.0286   0.0044          0.0266 10
```

Mixed Logit model

- As the linear mixed model is to the linear model, the mixed logit model is to ordinary logistic regression.
- We assume that individual differences within a grouping factor are normally distributed in the log-odds of the event of interest.

$$\ln \frac{\mathbf{p}}{\mathbf{1-p}} = \overbrace{\mathbf{X}\beta}^{\text{Fixed effects}} + \overbrace{\mathbf{Z}\mathrm{b}}^{\text{Random effects}} \quad , \quad \overbrace{\mathrm{b_i}}^{\text{Noise}\sim N(0,\sigma_{\mathrm{b_i}})}$$

Mixed logit model: a simple example

$$\ln \frac{p}{1-p} = \overbrace{\beta_0 + \beta_1 \textbf{NativeLanguage}}^{\text{Fixed effects}} + \overbrace{b_0}^{\text{Random intercept, } b_0 \sim N(0, \sigma_{b_0})}$$

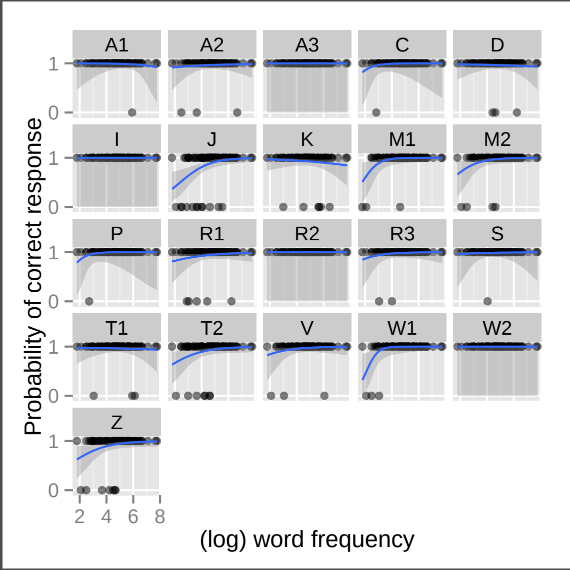
```
library(lme4)
l1 = lmer(Correct ~ 1 + NativeLanguage +
          (1 | Subject),
          data=lexdec, family=binomial)
l1

## Generalized linear mixed model fit by the Laplace approximation
## Formula: Correct ~ 1 + NativeLanguage + (1 | Subject)
##    Data: lexdec
##  AIC BIC logLik deviance
##  538 554   -266      532
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  Subject (Intercept) 0.525    0.724
## Number of obs: 1659, groups: Subject, 21
##
## Fixed effects:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)            3.694      0.293   12.59   <2e-16 ***
## NativeLanguageOther   -0.527      0.425   -1.24     0.22
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## NtvLnggOthr -0.690
```

Plotting by-grouping factor effects

- Sometimes it can be useful to visualize effects by grouping factor. The code shown below also works for other GLMs (just change the `family`.
- For example, to plot separate by-subject fits:

```
library(ggplot2)
ggplot(lexdec, aes(x=Frequency, y=ifelse(Correct == "correct", 1, 0))) +
  geom_point(alpha=.5) +
  geom_smooth(method='glm', family=binomial) +
  scale_y_discrete("Probability of correct response",
        limits=c(0,1)
  ) +
  scale_x_continuous("(log) word frequency") +
  coord_cartesian(ylim = c(-.1,1.1)) +
  facet_wrap(~ Subject)
```

Plotting by-grouping factor effects (cont'd)



Obtaining by-grouping factor regressions

- For example, to obtain coefficients for separate by-subject fitted logistic regression:

```
library(lme4)
l1 = lmList(Correct ~ Frequency | Subject, data = lexdec, family= binomial)
head(coef(l1), 8)

##    (Intercept)  Frequency
## A1       8.550 -7.847e-01
## A2       1.808  3.155e-01
## A3     -26.566  4.973e-18
## C       -1.220  1.510e+00
## D        4.313 -2.199e-01
## I      -26.566  4.972e-16
## J       -2.186  9.097e-01
## K        3.922 -2.877e-01
```

NB: These estimates, which do not assume normality of the between-subject differences, could be compared against the predicted effects of a GLMM, by using the fixed effect intercept and `Frequency` slope adjusted by their respective by-`Subject` BLUPs)

Quasi-binomial models

- Logistic regression assumes that the data is drawn from a binomial distribution.
- This entails that the mean of the distribution is assumed to determine the variance.

Quasi-binomial models

- Logistic regression assumes that the data is drawn from a binomial distribution.
- This entails that the mean of the distribution is assumed to determine the variance.
- However, this assumption can be questioned (though for binary data the assumption is almost always met). **Quasi-binomial** models allow us to fit a **dispersion** parameter.
- This dispersion tells us how close the ratio of mean and variance is to 1 (the assumed value in a binomial model)

Quasi-binomial models


```
qg= glm(Correct ~ 1 + NativeLanguage,
          data=lexdec, family=quasibinomial)
summary(qg)

##
## Call:
## glm(formula = Correct ~ 1 + NativeLanguage, family = quasibinomial,
##     data = lexdec)
##
## Deviance Residuals:
##    Min      1Q   Median      3Q     Max
## -2.654   0.245   0.245   0.327   0.327
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)            3.492      0.192   18.19   <2e-16 ***
## NativeLanguageOther   -0.590      0.256   -2.31    0.021 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasibinomial family taken to be 1.001)
##
##     Null deviance: 548.57  on 1658  degrees of freedom
## Residual deviance: 543.17  on 1657  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 6
```

Poisson models

For count data, we can use **Poisson** models:

- The linear predictor:

$$\eta = \alpha + \beta_1 \mathbf{x_1} + \cdots + \beta_n \mathbf{x_n}$$

- The link function g is the log-transform:

Poisson models

For count data, we can use **Poisson** models:

- The linear predictor:

$$\eta = \alpha + \beta_1 \mathbf{x_1} + \cdots + \beta_n \mathbf{x_n}$$

- The link function g is the log-transform:

$$\mathrm{E}(\mathbf{counts}) = \mu = \mathrm{g}^{-1}(\eta) \Leftrightarrow \quad (8)$$

$$\mathrm{g}(\mu) = \ln \mu = \alpha + \beta_1 \mathbf{x_1} + \cdots + \beta_n \mathbf{x_n} \Leftrightarrow \quad (9)$$

$$\mathrm{E}(\mathbf{counts}) = \exp(\mathbf{X}\beta) \quad (10)$$

- The distribution around the mean is taken to be Poisson.

Poisson regression (cont'd)

```
data(lexdec, package="languageR")
p = glm(round(exp(Frequency)) ~ 1 + NativeLanguage,
          data=lexdec, family=poisson)
summary(p)

##
## Call:
## glm(formula = round(exp(Frequency)) ~ 1 + NativeLanguage, family = poisson,
##     data = lexdec)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -21.04  -15.24   -9.47    2.18   80.24
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          5.52e+00   2.05e-03    2687   <2e-16 ***
## NativeLanguageOther -4.07e-16   3.14e-03       0        1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 586606  on 1658  degrees of freedom
## Residual deviance: 586606  on 1657  degrees of freedom
## AIC: 597546
##
## Number of Fisher Scoring iterations: 6
```

Mixed Poisson regression

```
data(lexdec, package="languageR")
p = lmer(round(exp(Frequency)) ~ 1 + NativeLanguage +
           (1 | Subject),
           data=lexdec, family=poisson)
summary(p)

## Generalized linear mixed model fit by the Laplace approximation
## Formula: round(exp(Frequency)) ~ 1 + NativeLanguage + (1 | Subject)
##    Data: lexdec
##     AIC    BIC  logLik deviance
##  586612 586628 -293303   586606
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  Subject (Intercept)  0        0
## Number of obs: 1659, groups: Subject, 21
##
## Fixed effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          5.52e+00   2.05e-03    2687   <2e-16 ***
## NativeLanguageOther -3.14e-05   3.14e-03       0     0.99
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## NtvLnggOthr -0.655
```

Ordinary Quasi-Poisson models

- Possion models, like binomial models, assume that the mean determines the variance of the noise distribution.
- For count data this might not hold.

- **Quasi-poisson** models allow us to fit a **dispersion** parameter.
- This dispersion tells us how close the ratio of mean and variance is to 1.

Quasi-Poisson models: an example


```
qp= glm(round(exp(Frequency)) ~ 1 + NativeLanguage,
           data=lexdec, family=quasipoisson)
summary(qp)
```

```
##
## Call:
## glm(formula = round(exp(Frequency)) ~ 1 + NativeLanguage, family = quasipoisson,
##     data = lexdec)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -21.04  -15.24   -9.47    2.18   80.24
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          5.52e+00   5.07e-02     109   <2e-16 ***
## NativeLanguageOther -4.07e-16   7.75e-02       0        1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 609.6)
##
##     Null deviance: 586606  on 1658  degrees of freedom
## Residual deviance: 586606  on 1657  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 6
```

Understanding dispersion

- In order to understand the concept of dispersion, it is useful to first get a better understanding of the Poisson distribution.
- The probability mass function of a **Poisson distribution** is:

$$P(Y = k) = \frac{\lambda^k \exp(-\lambda)}{k!} \quad (11)$$

...where k is a count and λ is the expected value (i.e., mean) for the random variable Y *and* its variance Y, i.e.:

$$\lambda = E(Y) = Var(Y) \quad (12)$$

- The Poisson distribution has a **dispersion index** of 1.

Understanding dispersion

- In order to understand the concept of dispersion, it is useful to first get a better understanding of the Poisson distribution.
- The probability mass function of a **Poisson distribution** is:

$$P(Y = k) = \frac{\lambda^k \exp(-\lambda)}{k!} \tag{11}$$

...where k is a count and λ is the expected value (i.e., mean) for the random variable Y *and* its variance Y, i.e.:

$$\lambda = E(Y) = \mathrm{Var}(Y) \tag{12}$$

- The Poisson distribution has a **dispersion index** of 1.
- Let's visualize the Poisson distribution for different means.

Visualizing the Poisson distribution

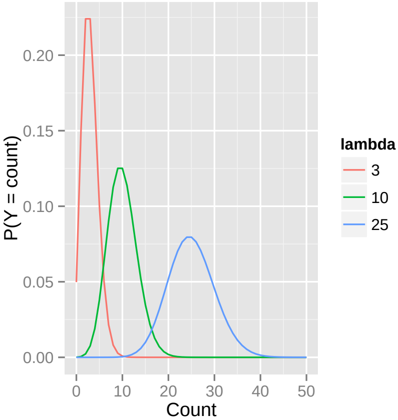
LI539 Mixed Effect Models

T. Florian Jaeger

Logit Models
Example
Ordinary
Mixed
Quasi-binomial
Ordinary
Poisson models
Ordinary
Mixed
Quasi-Poisson
Ordinary
Multivariate models
Multivariate Gaussians
Multinomial models
Ordered data

```
# R comes with a variety of build-in PMFs, CDFs, and sampling
# functions for, e.g. Gaussian, Binomial, Poisson, etc. distributions

# taken 5 samples from a poisson with mean = 7
rpois(n=5, lambda=7)

## [1]  6 10  5  9  3


library(ggplot2)
d = data.frame(x = rep(0:50, 3),
               lambda = sort(rep(c(3,10,25), 51)))
d$y = with(d, dpois(x, lambda))
```

Visualizing the Poisson distribution (cont'd)



- Notice how the variance of the distribution increases with the mean.

Poisson vs. actual count distribution

- Now, we're ready to illustrate what dispersion means.
- Let's take an even simpler model than above and only fit an intercept:

```
s1 = summary(glm(round(exp(Frequency)) ~ 1,
          data=lexdec, family=poisson))
s1

##
## Call:
## glm(formula = round(exp(Frequency)) ~ 1, family = poisson, data = lexdec)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -21.04  -15.24   -9.47    2.18   80.24
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.52070    0.00155    3554   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 586606  on 1658  degrees of freedom
## Residual deviance: 586606  on 1658  degrees of freedom
## AIC: 597544
##
## Number of Fisher Scoring iterations: 6

# the intercept is the predicted mean (log) count
# let's compare the actual distribution for this mean
# against what would be expected under a Poisson.
```

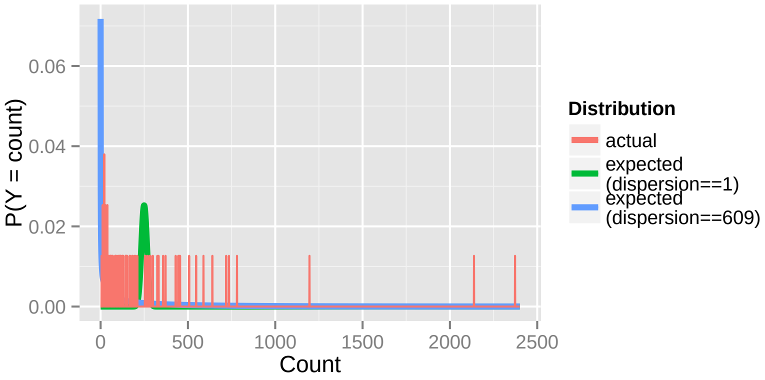
Poisson vs. actual count distribution (cont'd)


```
s2 = summary(glm(round(exp(Frequency)) ~ 1,
           data=lexdec, family=quasipoisson))
s2

##
## Call:
## glm(formula = round(exp(Frequency)) ~ 1, family = quasipoisson,
##     data = lexdec)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -21.04  -15.24   -9.47    2.18   80.24
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.5207     0.0383     144   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 609.2)
##
##     Null deviance: 586606  on 1658  degrees of freedom
## Residual deviance: 586606  on 1658  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 6
```

Poisson vs. actual count distribution (cont'd)


```r
# create data.frame with distribution values
d = data.frame(x = rep(0:2400,3),
               y.value = c(rep("expected\n(dispersion==1)", 2401),
                           rep("expected\n(dispersion==609)", 2401),
                           rep("actual", 2401))
)

dqpois = function(x, mu, theta) {
  dnbinom(x = x, mu = mu, size = mu/(theta-1))
}
d$y = ifelse(d$y.value == "expected\n(dispersion==1)",
             with(d, dpois(x,
                           lambda = exp(s1$coefficients[,1]))),
          ifelse(d$y.value == "expected\n(dispersion==609)",
             with(d, dqpois(x,
                            mu = exp(s2$coefficients[,1]),
                            theta = 609)),
             (table(round(exp(lexdec$Frequency))) / 21)[as.character(d$x)]))
d$y = ifelse(is.na(d$y), 0, d$y)
```

Visualization



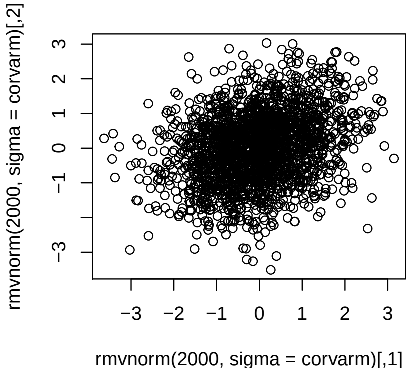
Multivariate GLMMs

- **Multivariate GLMMs** allow the simultaneous analysis of multiple dependent variable.
- The primary advantage of multivariate analyses over separate univariate analyses is that the multivariate analysis allows us to recognize (and correct for) covariant between the dependent variables.
- These models are, however, technically quite challenging and currently aren't easily done within the framework of `lme4` (but see `http://rstudio-pubs-static.s3.amazonaws.com/3336_03636030d93d47de9131e625b72f58c6.html`)
  - A relatively common example are multiple continuous dependent variables
  - **MANOVA** is a common method that has been used to analyze such data sets.
  - **Multivariate Gaussian models** are an alternative that allow us to analyze such data sets within the **GLM** framework.
  - **Multivariate Mixed Gaussian models** provide an alternative within the **GLMM** framework (i.e., a framework that allows the inclusion of random effects to account for clusters in the data).

Multivariate normal distributions

- What is a multivariate distribution?
- Here is an example from a bivariate normal distribution:

```
library(mvtnorm)
# create a variance-covariance matrix for a bivariate normal
# distribution with a mild correlation of the two variances
corvarm = matrix(c(1,.3,.3,1), ncol=2)
plot(rmvnorm(2000, sigma=corvarm))
```



Ordinary multivariate Gaussian regression

- The package `VGAM` provides functions for multivariate **GLM** and **GAM** models (Generalized Additive Models, which allow to model locally non-linear models)
- Here, I provide an example analysis of a bivariate Gaussian outcome, using `vglm()` from the `VGAM` package:

```
library(VGAM)
data(lexdec, package="languageR")
v = vglm(cbind(Frequency, FamilySize) ~ Length,
     family = gaussianff,
     data = lexdec)

## [1] "head(wz) ------------------------------"
##       PriorWeight1 PriorWeight2
## [1,]             1            1
## [2,]             1            1
## [3,]             1            1
## [4,]             1            1
## [5,]             1            1
## [6,]             1            1
## [1] "hi3 ooooo"
```

Ordinary multivariate Gaussian regression (cont'd)


```
##
## Call:
## vglm(formula = cbind(Frequency, FamilySize) ~ Length, family = gaussianff,
##     data = lexdec)
##
## Pearson Residuals:
##              Min    1Q Median   3Q Max
## Frequency  -2.4 -0.71  0.098 0.82 2.8
## FamilySize -1.3 -0.38 -0.082 0.22 2.4
##
## Coefficients:
##               Estimate Std. Error z value
## (Intercept):1     6.49      0.077      84
## (Intercept):2     2.46      0.077      32
## Length:1         -0.29      0.012     -24
## Length:2         -0.30      0.012     -24
##
## Number of linear predictors:  2
##
## Names of linear predictors: Frequency, FamilySize
##
## (Estimated) Dispersion Parameter for gaussianff family:   0.89
##
## Residual deviance: 2951 on 3314 degrees of freedom
##
## Log-likelihood: -4524 on 3314 degrees of freedom
##
## Number of iterations: 2
```

Proportional odds

- The **proportional odds** or **ordered logistic regression** model can be thought of as an extension of binary logistic regression.
- The proportional odds model can be used to analyze ordered multinomial (i.e., ordinal) outcomes, such as a 5-way acceptability rating on a scale like 'perfectly unacceptable', 'somewhat unacceptable', 'don't know', 'somewhat acceptable', 'perfectly acceptable'.
- The proportional odds model does *not* assume that the levels of the outcome are equidistant.
  - By default, the model will fit $k - 1$ intercept terms for k outcome levels (if no level order is specified, the alphanumerically first level will serve as baseline).
  - These intercepts predict the log-odds of the different outcome levels
- However, the effects of the other predictors are assumed to be constant in (log)odds space across all levels.

Proportional odds model in rms


```
library(rms)
data(lexdec, package="languageR")

# create fake ordinal outcome out of RT variable, by
# cutting it into four bins
lexdec$ORT = cut(lexdec$RT, 4,
    labels= c('lowest RT','low RT','high RT','highest RT'))

# how are data distributed
summary(lexdec$ORT)

##   lowest RT      low RT     high RT highest RT
##         552         927         170         10
```

Proportional odds model in rms (cont'd)


```
lexdec$cFrequency = lexdec$Frequency - mean(lexdec$Frequency)
lexdec$NativeLanguage =
  factor(lexdec$NativeLanguage, levels=c('Other', 'English'))
contrasts(lexdec$NativeLanguage) = cbind('English' = c(-.5,.5))

po = lrm(ORT ~ 1 + cFrequency + NativeLanguage,
           data=lexdec, x=T, y=T)
```

Proportional odds model in rms (cont'd)


```
# some lrm() specific setup
dd = datadist(lexdec)
options(datadist='dd')

po

##
## Logistic Regression Model
##
## lrm(formula = ORT ~ 1 + cFrequency + NativeLanguage, data = lexdec,
##     x = T, y = T)
##
## Frequencies of Responses
##
##  lowest RT      low RT    high RT highest RT
##        552         927        170         10
##
##                        Model Likelihood     Discrimination    Rank Discrim.
##                           Ratio Test           Indexes          Indexes
## Obs           1659    LR chi2     238.35    R2       0.157    C       0.694
## max |deriv| 4e-14     d.f.             2    g        0.914    Dxy     0.388
##                       Pr(> chi2) <0.0001    gr       2.493    gamma   0.393
##                                             gp       0.084    tau-a   0.220
##                                             Brier    0.087
##
##                           Coef    S.E.   Wald Z Pr(>|Z|)
## y>=low RT                  0.9110 0.0584  15.59 <0.0001
## y>=high RT                -2.2337 0.0833 -26.81 <0.0001
## y>=highest RT             -5.3243 0.3189 -16.70 <0.0001
## cFrequency                -0.3536 0.0400  -8.83 <0.0001
## NativeLanguage=English -1.3240 0.1066 -12.42 <0.0001
```