- Binary logistic regression, also called a logit model, is used to model dichotomous outcome variables. In the logit model the log odds of the outcome is modeled as a linear combination of the predictor variables.
- Multinomial logistic regression is used to model nominal outcome variables. Again the log odds of the outcomes are modeled as a linear combination of the predictor variables.
- Ordinal logistic regression is used to model nominal outcome variables, where a hierarchy of categories exists.

- Poisson regression is used to model count variables.
- Negative binomial regression is for modeling count variables, usually for over-dispersed count outcome variables.

Multinomial Logistic Regression

- Multinomial logistic regression is used to model nominal outcome variables, in which the log odds of the outcomes are modeled as a linear combination of the predictor variables.

Multinomial Regression with R

The main package we will use is the **nnet** package. We will also use the **ggplot2** and **reshape2** package.

```
install.packages("nnet")
library(nnet)

library(ggplot2)
library(reshape2)
```

Multinomial Regression with R

Examples of multinomial logistic regression

Example 1. Entering high school students make program choices among general program, vocational program and academic program. Their choice might be modeled using their writing score and their social economic status.

Multinomial Regression with R

Examples of multinomial logistic regression

Example 2. People's occupational choices might be influenced by their parents' occupations and their own education level. We can study the relationship of one's occupation choice with education level and father's occupation. The occupational choices will be the outcome variable which consists of categories of occupations.

Multinomial Regression with R

Examples of multinomial logistic regression

Example 3. A biologist may be interested in food choices that alligators make. Adult alligators might have different preferences from young ones. The outcome variable here will be the types of food, and the predictor variables might be size of the alligators and other environmental variables.

Description of the Data

We will use the third example using the *multilog* data set (or *ml*).

```
ml <- read.csv("multilog.csv",header=T)
```

Description of the Data

- The outcome variable is **prog**, program type.
- The two predictor variables are

1. social economic status, **ses**, (a three-level categorical variable)
2. writing score, **write**, (a continuous variable).

- The data set contains variables on 200 students.

Multinomial Regression with R

```
table(ml$ses, ml$prog)

       general academic vocation
low         16       19       12
middle      20       44       31
high         9       42        7

               M    SD
 general   51.33 9.398
 academic 56.26 7.943
 vocation 46.76 9.319
```

Multinomial logistic regression

- We use the `multinom` function from the **nnet** package to estimate a multinomial logistic regression model.
- Remark - There are other functions in other R packages capable of multinomial regression, such as the **mlogit** package.
- The multinom function does not require the data to be reshaped (as the mlogit package does) and to mirror the example code found in Hilbe's Logistic Regression Models.

Multinomial logistic regression

- We must choose the level of our outcome that we wish to use as our **baseline** and specify this in the `relevel` function.
  Let's choose "academic".
- Then, we run our model using *multinom*.
- The multinom command does not include p-value calculation for the regression coefficients, but we calculate p-values using Wald tests (here z-tests).

```
ml$prog2 <- relevel(ml$prog, ref = "academic")
test <- multinom(prog2 ~ ses + write, data = ml)

 # weights:  15 (8 variable)
 initial  value 219.722458
 iter  10 value 179.982880
 final  value 179.981726
 converged
```

Multinomial Regression with R

```
summary(test)

 Call:
 multinom(formula = prog2 ~ ses + write,
   data = ml)

 Coefficients:
          (Intercept) sesmiddle seshigh     write
 general        2.852   -0.5333 -1.1628 -0.05793
 vocation       5.218    0.2914 -0.9827 -0.11360
```

Multinomial Regression with R

```
....
 Std. Errors:
          (Intercept) sesmiddle seshigh   write
 general        1.166    0.4437  0.5142 0.02141
 vocation       1.164    0.4764  0.5956 0.02222

 Residual Deviance: 360
 AIC: 376
```

Multinomial Regression with R

```
# 2-tailed z test
# p.values

          (Intercept) sesmiddle seshigh     write
 general    1.448e-02    0.2294 0.02374 6.819e-03
 vocation   7.299e-06    0.5408 0.09895 3.176e-07
```

Multinomial Regression with R

- Some output is generated by running the model, even though we are assigning the model to a new R object.
- This model-running output includes some iteration history and includes the final negative log-likelihood 179.981726.
- This value multiplied by two is then seen in the model summary as the **Residual Deviance** and it can be used in comparisons of nested models.

Multinomial Regression with R

- As with many summary outputs, the output contains a column of coefficients and a column of standard errors.
- Each of these blocks has one row of values corresponding to a model equation.
- Focusing on the block of coefficients, we can look at the first row comparing `prog = "general"` to our baseline `prog = "academic"` and the second row comparing `prog = "vocation"` to our baseline `prog = "academic"`.

Multinomial Regression with R

- If we consider our coefficients from the first row to be b_1 and our coefficients from the second row to be b_2, we can write our model equations:

$$ln\left(\frac{P(prog = gen.)}{P(prog = acad.)}\right) = b_{10} + b_{11}(ses = 2) + b_{12}(ses = 3) + b_{13}write$$

$$ln\left(\frac{P(prog = voc.)}{P(prog = acad.)}\right) = b_{20} + b_{21}(ses = 2) + b_{22}(ses = 3) + b_{23}write$$

Multinomial Regression with R

- A one-unit increase in the variable **write** is associated with the decrease in the log odds of being in general program vs. academic program in the amount of 0.058 (b_{13}).
- A one-unit increase in the variable **write** is associated with the decrease in the log odds of being in vocation program vs. academic program in the amount of o.1136 (b_{23}).

Multinomial Regression with R

- The log odds of being in general program vs. in academic program will decrease by 1.163 if moving from ses="low" to ses="high"(b_12).
- The log odds of being in general program vs. in academic program will decrease by 0.533 if moving from ses="low" to ses="middle"(b_11), although this coefficient is not significant.

Multinomial Regression with R

- The log odds of being in vocation program vs. in academic program will decrease by 0.983 if moving from ses="low" to ses="high"(b_22).
- The log odds of being in vocation program vs. in academic program will increase by 0.291 if moving from ses="low" to ses="middle"(b_21), although this coefficient is not signficant.

Multinomial Logistic Regression with R

- The ratio of the probability of choosing one outcome category over the probability of choosing the baseline category is often referred as **relative risk**
- It is also sometimes referred as **odds**

Multinomial Logistic Regression with R

- The relative risk is the (right-hand side) linear equation exponentiated, leading to the fact that the exponentiated regression coefficients are relative risk ratios for a unit change in the predictor variable.
- We can exponentiate the coefficients from our model to see these risk ratios (next slide).

Multinomial Logistic Regression with R

Extract the coefficients from the model then and exponentiate

```
exp(coef(test))

          (Intercept) sesmiddle seshigh  write
 general        17.33    0.5867  0.3126 0.9437
 vocation      184.61    1.3383  0.3743 0.8926
```

Multinomial Regression with R

- The relative risk ratio for a one-unit increase in the variable **write** is 0.9437 for being in general program vs. academic program.
- The relative risk ratio switching from **ses** $= 1$ to 3 is 0.3126 for being in general program vs. academic program.

Multinomial Regression with R

- You can also use predicted probabilities to help you understand the model.
- You can calculate predicted probabilities for each of our outcome levels using the `fitted` function.
- We can start by generating the predicted probabilities for the observations in our dataset and viewing the first few rows

Multinomial Regression with R

```
head(pp <- fitted(test))

  academic general vocation
1   0.1483  0.3382   0.5135
2   0.1202  0.1806   0.6992
3   0.4187  0.2368   0.3445
4   0.1727  0.3508   0.4765
5   0.1001  0.1689   0.7309
6   0.3534  0.2378   0.4088
```

Multinomial Regression with R

Prediction

- Suppose we want to examine the changes in predicted probability associated with one of our two variables, we can create small datasets varying one variable while holding the other constant.
- We will first do this holding **write** at its mean and examining the predicted probabilities for each level of **ses**.
- Three Cases

Multinomial Logistic Regression with R

```
dses <- data.frame(ses = c("low", "middle", "h
predict(test, newdata = dses, "probs")

   academic general vocation
 1   0.4397  0.3582   0.2021
 2   0.4777  0.2283   0.2939
 3   0.7009  0.1785   0.1206
```

Multinomial Logistic Regression with R

- Another way to understand the model using the predicted probabilities is to look at the averaged predicted probabilities for different values of the continuous predictor variable **write** within each level of **ses**.

Multinomial Logistic Regression with R

Store the predicted probabilities for each value of ses and write

```
dwrite <- data.frame(
  ses = rep(c("low", "middle", "high"), each = 41),
  write = rep(c(30:70), 3))
```

```
pp.write <- cbind(dwrite,
     predict(test, newdata = dwrite,
     type = "probs", se = TRUE))
```

Multinomial Logistic Regression with R

Calculate the mean probabilities within each level of ses

```
by(pp.write[, 3:5], pp.write$ses, colMeans)
```

```
 pp.write$ses: high
 academic   general  vocation
   0.6164    0.1808    0.2028
 -------------------------------------
 pp.write$ses: low
 academic   general  vocation
   0.3973    0.3278    0.2749
 -------------------------------------
 pp.write$ses: middle
 academic   general  vocation
   0.4256    0.2011    0.3733
```

Multinomial Regression with R

- A couple of plots can convey a good deal amount of information.
- Using the predictions we generated for the `pp.write` object above, we can plot the predicted probabilities against the writing score by the level of **ses** for different levels of the outcome variable.

Multinomial Logistic Regression with R

remark: melt data set to long for ggplot2

```
lpp <- melt(pp.write, id.vars = c("ses", "write"),
            value.name = "probability")

head(lpp)  # view first few rows

  ses write variable probability
1 low    30 academic     0.09844
2 low    31 academic     0.10717
3 low    32 academic     0.11650
4 low    33 academic     0.12646
5 low    34 academic     0.13705
6 low    35 academic     0.14828
```

Multinomial Regression with R

plot predicted probabilities across write values for each level of ses facetted by program type

```
ggplot(lpp, aes(x = write, y = probability, co
    ., scales = "free")
```

Multinomial Regression with R

Things to consider

- The Independence of Irrelevant Alternatives (IIA) assumption: Roughly, the IIA assumption means that adding or deleting alternative outcome categories does not affect the odds among the remaining outcomes.
- There are alternative modeling methods, such as alternative-specific multinomial probit model, or nested logit model to relax the IIA assumption.

Multinomial Regression with R

Things to consider

- Diagnostics and model fit: Unlike logistic regression where there are many statistics for performing model diagnostics, it is not as straightforward to do diagnostics with multinomial logistic regression models.

Multinomial Regression with R

Things to consider

- For the purpose of detecting outliers or influential data points, one can run separate logit models and use the diagnostics tools on each model.
- Sample size: Multinomial regression uses a maximum likelihood estimation method, it requires a large sample size. It also uses multiple equations. This implies that it requires an even larger sample size than ordinal or binary logistic regression.

Multinomial Regression with R

Things to consider

- Complete or quasi-complete separation: Complete separation means that the outcome variable separate a predictor variable completely, leading perfect prediction by the predictor variable.

Multinomial Regression with R

Things to consider

- Perfect prediction means that only one value of a predictor variable is associated with only one value of the response variable. But you can tell from the output of the regression coefficients that something is wrong. You can then do a two-way tabulation of the outcome variable with the problematic variable to confirm this and then rerun the model without the problematic variable.

Multinomial Regression with R

Things to consider

- Empty cells or small cells: You should check for empty or small cells by doing a cross-tabulation between categorical predictors and the outcome variable. If a cell has very few cases (a small cell), the model may become unstable or it might not even run at all.

Ordered logistic regression R

`polr`

- In this section we will use the `polr` command (from the MASS package) to estimate an ordered logistic regression model.
- The command name comes from **proportional odds logistic regression**, due to the the **proportional odds assumption** in the model.

Ordered logistic regression `R`

`polr`

- `polr` uses the standard formula interface in `R` for specifying a regression model with outcome followed by predictors.
- We will also specify `Hess=TRUE` to have the model return the observed information matrix from optimization (called the Hessian) which is used to get standard errors.

Ordered logistic regression R

```
## fit ordered logit model and store results 'm'
m <- polr(apply ~ pared +
          public + gpa, data = dat, Hess=TRUE)

## view a summary of the model
summary(m)
## Call:
## polr(formula = apply ~ pared +
             public + gpa, data = dat, Hess = TRUE)
```

Ordered logistic regression R

```
Coefficients:
         Value Std. Error t value
pared   1.0477      0.266   3.942
public -0.0588      0.298  -0.197
gpa     0.6159      0.261   2.363

Intercepts:
                            Value  Std. Error t value
unlikely|somewhat likely     2.204  0.780       2.827
somewhat likely|very likely  4.299  0.804       5.345
```

Ordered logistic regression R

1 The “Call”, what type of model we ran, what options we specified, etc.

2 The usual regression output coefficient table including the value of each coefficient, standard errors, and $t-$value, which is simply the ratio of the coefficient to its standard error.
(Remark: There is no significance test by default.)

Ordered logistic regression R

3 We then have the estimates for the two intercepts, which are sometimes called **cutpoints**.

4 The intercepts indicate where the latent variable is cut to make the three groups that we observe in our data.

Ordered logistic regression R

- Note that this latent variable is continuous. In general, these are not used in the interpretation of the results.
- The cutpoints are closely related to thresholds, which are reported by other statistical packages.

Ordered logistic regression R

- Finally, we see the residual deviance, -2 * Log Likelihood of the model as well as the AIC.
- Both the deviance and AIC are useful for model comparison.
- Some people are not satisfied without a $p-$value.
- One way to calculate a p-value in this case is by comparing the t-value against the standard normal distribution, like a $z-$test.

Ordered logistic regression R

- Of course this is only true with infinite degrees of freedom, but is reasonably approximated by large samples, becoming increasingly biased as sample size decreases.
- First we store the coefficient table, then calculate the $p-$values and combine back with the table.

Ordered logistic regression R

```
# store table
(ctable <- coef(summary(m)))
                              Value Std. Error t value
 pared                      1.04769     0.2658  3.9418
 public                    -0.05879     0.2979 -0.1974
 gpa                        0.61594     0.2606  2.3632
 unlikely|somewhat likely   2.20391     0.7795  2.8272
 somewhat likely|very likely 4.29936    0.8043  5.3453
```

Ordered Logistic regression R

```
# calculate and store p values
p <- pnorm(abs(ctable[, "t value"]),
      lower.tail = FALSE) * 2

# Combined table
(ctable <- cbind(ctable, "p value" = p))
                      Value Std. Error t value   p value
 pared              1.04769     0.2658  3.9418 8.087e-05
 public            -0.05879     0.2979 -0.1974 8.435e-01
 gpa                0.61594     0.2606  2.3632 1.812e-02
 unli..|some..      2.20391     0.7795  2.8272 4.696e-03
 some..|very..      4.29936     0.8043  5.3453 9.027e-08
```

Ordered Logistic Regression R

- We can also get confidence intervals for the parameter estimates.
- These can be obtained either by profiling the likelihood function or by using the standard errors and assuming a normal distribution.
- Note that profiled CIs are not symmetric (although they are usually close to symmetric).
- If the 95% CI does not cross 0, the parameter estimate is statistically significant.

Ordered Logistic Regression R

```
(ci <- confint(m))
# default method gives profiled CIs

 Waiting for profiling to be done...
            2.5 %   97.5 %
 pared    0.5282   1.5722
 public -0.6522   0.5191
 gpa      0.1076   1.1309
```

Ordered Logistic Regression R

```
confint.default(m) # CIs assuming normality

          2.5 %   97.5 %
pared    0.5268   1.569
public  -0.6426   0.525
gpa      0.1051   1.127
```

Ordered Logistic Regression R

Confidence Intervals

- The CIs for both pared and gpa do not include 0; but the CI for public does.
- The estimates in the output are given in units of ordered logits, or ordered log odds.
- So for pared, we would say that for a one unit increase in pared (i.e., going from 0 to 1), we expect a 1.05 increase in the expect value of apply on the log odds scale, given all of the other variables in the model are held constant.

Ordered Logistic Regression R

Confidence Intervals

- For gpa, we would say that for a one unit increase in gpa, we would expect a 0.62 increase in the expected value of apply in the log odds scale, given that all of the other variables in the model are held constant.

Ordered Logistic Regression R

- The coefficients from the model can be somewhat difficult to interpret because they are scaled in terms of logs.
- Another way to interpret logistic regression models is to convert the coefficients into odds ratios.
- To get the Odds Ratios and confidence intervals, we just exponentiate the estimates and confidence intervals.

Ordered Logistic Regression R

Odds Ratios

```
exp(coef(m))
  pared public    gpa
 2.8511 0.9429 1.8514
```

```
 # Odds Ratios and CIs
exp(cbind(OR = coef(m), ci))
            OR  2.5 % 97.5 %
 pared  2.8511 1.6958  4.817
 public 0.9429 0.5209  1.681
 gpa    1.8514 1.1136  3.098
```

Ordered Logistic Regression R

- These coefficients are called **proportional odds ratios** and we would interpret these pretty much as we would odds ratios from a binary logistic regression.
- For pared, we would say that for a one unit increase in parental education, i.e., going from 0 (*Low*) to 1 (*High*), the odds of "*very likely*" applying versus "*somewhat likely*" or "*unlikely*" applying combined are 2.85 greater, given that all of the other variables in the model are held constant.

Ordered Logistic Regression R

- Similarly, the odds "*very likely*" or "*somewhat likely*" applying versus "*unlikely*" applying is 2.85 times greater, given that all of the other variables in the model are held constant.
- For gpa (and other continuous variables), the interpretation is that when a student's gpa moves 1 unit, the odds of moving from "*unlikely*" applying to "*somewhat likely*" or "*very likely*" applying (or from the lower and middle categories to the high category) are multiplied by 1.85.

Ordinal Logistic Regression with R

One of the assumptions underlying ordinal logistic (and ordinal probit) regression is that the relationship between each pair of outcome groups is the same. In other words, ordinal logistic regression assumes that the coefficients that describe the relationship between, say, the lowest versus all higher categories of the response variable are the same as those that describe the relationship between the next lowest category and all higher categories, etc. This is called the proportional odds assumption or the parallel regression assumption.

Ordinal Logistic Regression with R

Because the relationship between all pairs of groups is the same, there is only one set of coefficients. If this was not the case, we would need different sets of coefficients in the model to describe the relationship between each pair of outcome groups. Thus, in order to asses the appropriateness of our model, we need to evaluate whether the proportional odds assumption is tenable.

Ordinal Logistic Regression with R

Statistical tests to do this are available in some software packages. However, these tests have been criticized for having a tendency to reject the null hypothesis (that the sets of coefficients are the same), and hence, indicate that there the parallel slopes assumption does not hold, in cases where the assumption does hold (see Harrell 2001 p. 335). We were unable to locate a facility in R to perform any of the tests commonly used to test the parallel slopes assumption.

Ordinal Logistic Regression with R

However, Harrell does recommend a graphical method for assessing the parallel slopes assumption. The values displayed in this graph are essentially (linear) predictions from a logit model, used to model the probability that y is greater than or equal to a given value (for each level of y), using one predictor (x) variable at a time. In order create this graph, you will need the Hmisc library.

Ordinal Logistic Regression with R

The code below contains two commands (the first command falls on multiple lines) and is used to create this graph to test the proportional odds assumption. Basically, we will graph predicted logits from individual logistic regressions with a single predictor where the outcome groups are defined by either apply $>=2$ and apply $>=3$.

Ordinal Logistic Regression with R

If the difference between predicted logits for varying levels of a predictor, say pared, are the same whether the outcome is defined by apply $>=$ 2 or apply $>=$3, then we can be confident that the proportional odds assumption holds. In other words, if the difference between logits for pared $=$ 0 and pared $=$ 1 is the same when the outcome is apply $>=$ 2 as the difference when the outcome is apply $>=$ 3, then the proportional odds assumption likely holds.

Ordinal Logistic Regression with R

The first command creates the function that estimates the values that will be graphed. The first line of this command tells R that sf is a function, and that this function takes one argument, which we label y. The sf function will calculate the log odds of being greater than or equal to each value of the target variable. For our purposes, we would like the log odds of apply being greater than or equal to 2, and then greater than or equal to 3.

Ordinal Logistic Regression with R

Depending on the number of categories in your dependent variable, and the coding of your variables, you may have to edit this function. Below the function is configured for a y variable with three levels, 1, 2, 3. If your dependent variable has 4 levels, labeled 1, 2, 3, 4 you would need to add 'Y>=4'=qlogis(mean(y >= 4)) (minus the quotation marks) inside the first set of parentheses. If your dependent variable were coded 0, 1, 2 instead of 1, 2, 3, you would need to edit the code, replacing each instance of 1 with 0, 2 with 1, and so on. Inside the sf function we find the qlogis function, which transforms a probability to a logit.

Ordinal Logistic Regression with R

So, we will basically feed probabilities of apply being greater than 2 or 3 to qlogis, and it will return the logit transformations of these probabilites. Inside the qlogis function we see that we want the log odds of the mean of y $>=$ 2. When we supply a y argument, such as apply, to function sf, y $>=$ 2 will evaluate to a 0/1 (FALSE/TRUE) vector, and taking the mean of that vector will give you the proportion of or probability that apply $>=$ 2.

Ordinal Logistic Regression with R

- The second command below calls the function sf on several subsets of the data defined by the predictors.
- In this statement we see the summary function with a formula supplied as the first argument.
- When R sees a call to summary with a formula argument, it will calculate descriptive statistics for the variable on the left side of the formula by groups on the right side of the formula and will return the results in a nice table.

Ordinal Logistic Regression with R

By default, summary will calculate the mean of the left side variable. So, if we had used the code summary(as.numeric(apply)   pared + public + gpa) without the fun argument, we would get means on apply by pared, then by public, and finally by gpa broken up into 4 equal groups. However, we can override calculation of the mean by supplying our own function, namely sf to the fun= argument. The final command asks R to return the contents to the object s, which is a table.

Ordered Logistic regression R

Turning our attention to the predictions with public as a predictor variable, we see that when public is set to "no" the difference in predictions for apply greater than or equal to two, versus apply greater than or equal to three is about 2.14 (-0.204 - -2.345 = 2.141). When public is set to "yes" the difference between the coefficients is about 1.37 (-0.175 - -1.547 = 1.372).

Ordered Logistic regression R

The differences in the distance between the two sets of coefficients (2.14 vs. 1.37) may suggest that the parallel slopes assumption does not hold for the predictor public. That would indicate that the effect of attending a public versus private school is different for the transition from "unlikely" to "somewhat likely" and "somewhat likely" to "very likely."

Ordered Logistic regression R

The plot command below tells R that the object we wish to plot is s. The command which=1:3 is a list of values indicating levels of y should be included in the plot. If your dependent variable had more than three levels you would need to change the 3 to the number of categories (e.g. 4 for a four category variable, even if it is numbered 0, 1, 2, 3). The command pch=1:3 selects the markers to use, and is optional, as are xlab='logit' which labels the x-axis, and main=' ' which sets the main label for the graph to blank.

Ordered logistic regression R

If the proportional odds assumption holds, for each predictor variable, distance between the symbols for each set of categories of the dependent variable, should remain similar. To help demonstrate this, we normalized all the first set of coefficients to be zero so there is a common reference point.

Ordered logistic regression R

Looking at the coefficients for the variable pared we see that the distance between the two sets of coefficients is similar. In contrast, the distances between the estimates for public are different (i.e. the markers are much further apart on the second line than on the first), suggesting that the proportional odds assumption may not hold.

Ordered logistic regression R

```
plot(s, which=1:3, pch=1:3,
    xlab='logit', main=' ',
    xlim=range(s[,3:4]))
```

Ordered logistic regression R

- Once we are done assessing whether the assumptions of our model hold, we can obtain predicted probabilities, which are usually easier to understand than either the coefficients or the odds ratios.
- For example, we can vary gpa for each level of pared and public and calculate the probability of being in each category of apply.
- We do this by creating a new dataset of all the values to use for prediction.

Ordered logistic regression R

```
newdat <- data.frame(
  pared = rep(0:1, 200),
  public = rep(0:1, each = 200),
  gpa = rep(seq(from = 1.9, to = 4,
      length.out = 100), 4))

newdat <- cbind(newdat, predict(m,
   newdat, type = "probs"))
```

Ordered logistic regression R

```
# Show first few rows
head(newdat)
  pared public   gpa unlikely somewhat likely very likely
1     0      0 1.900   0.7376          0.2205      0.04192
2     1      0 1.921   0.4932          0.3946      0.11221
3     0      0 1.942   0.7325          0.2245      0.04299
4     1      0 1.964   0.4867          0.3985      0.11484
5     0      0 1.985   0.7274          0.2285      0.04407
6     1      0 2.006   0.4802          0.4023      0.11753
```

Ordered logistic regression R

- Now we can reshape the data long with the reshape2 package and plot all of the predicted probabilities for the different conditions.
- We plot the predicted probilities, connected with a line, coloured by level of the outcome, apply, and facetted by level of pared and public.
- We also use a custom label function, to add clearer labels showing what each column and row of the plot represent.

Ordered logistic regression R

```
library(reshape2)

lnewdat <- melt(newdat,
  id.vars = c("pared", "public", "gpa"),
  variable.name = "Level",
  value.name="Probability")
```

Ordered logistic regression R

```
%## view first few rows
%head(lnewdat)
%##   pared public   gpa    Level Probability
%## 1     0      0 1.900 unlikely      0.7376
%## 2     1      0 1.921 unlikely      0.4932
%## 3     0      0 1.942 unlikely      0.7325
%## 4     1      0 1.964 unlikely      0.4867
%## 5     0      0 1.985 unlikely      0.7274
%## 6     1      0 2.006 unlikely      0.4802
```

Ordered logistic regression R

```
ggplot(lnewdat, aes(x = gpa, y = Probability,
  colour = Level)) +
  geom_line() +
  facet_grid(pared ~ public, scales="free",
    labeller=function(x, y) sprintf("%s = %d", x, y))
```

Ordered logistic regression R

Things to consider
Perfect prediction: Perfect prediction means that one value of a predictor variable is associated with only one value of the response variable.

Ordered logistic regression R

Sample size: Both ordered logistic and ordered probit, using maximum likelihood estimates, require sufficient sample size.
Empty cells or small cells: You should check for empty or small cells by doing a crosstab between categorical predictors and the outcome variable. If a cell has very few cases, the model may become unstable or it might not run at all.

Ordered logistic regression R

Pseudo-R-squared: There is no exact analog of the R-squared found in OLS. There are many versions of pseudo-R-squares. Please see Long and Freese 2005 for more details and explanations of various pseudo-R-squares.
Diagnostics: Doing diagnostics for non-linear models is difficult, and ordered logit/probit models are even more difficult than binary models.

Poisson regression is used to model count variables.
This page uses the following packages. Make sure that you can load them before trying to run the examples on this page. If you do not have a package installed, run: install.packages("packagename"), or if you see the version is out of date, run: update.packages().

Poisson Regression with R

Please note: The purpose of this page is to show how to use various data analysis commands. It does not cover all aspects of the research process which researchers are expected to do. In particular, it does not cover data cleaning and checking, verification of assumptions, model diagnostics or potential follow-up analyses.

Examples of Poisson regression

Example 1. The number of persons killed by mule or horse kicks in the Prussian army per year. Ladislaus Bortkiewicz collected data from 20 volumes of Preussischen Statistik. These data were collected on 10 corps of the Prussian army in the late 1800s over the course of 20 years.

Example 2. The number of people in line in front of you at the grocery store. Predictors may include the number of items currently offered at a special discounted price and whether a special event (e.g., a holiday, a big sporting event) is three or fewer days away.

Example 3. The number of awards earned by students at one high school. Predictors of the number of awards earned include the type of program in which the student was enrolled (e.g., vocational, general or academic) and the score on their final exam in math.

Poisson Regression with R

Description of the data
For the purpose of illustration, we have simulated a data set for Example 3 above. In this example, num_awards is the outcome variable and indicates the number of awards earned by students at a high school in a year, math is a continuous predictor variable and represents students' scores on their math final exam, and prog is a categorical predictor variable with three levels indicating the type of program in which the students were enrolled. It is coded as 1 = "General", 2 = "Academic" and 3 = "Vocational". Let's start with loading the data and looking at some descriptive statistics.

Poisson Regression with R

```
p <- read.csv("http://www.ats.ucla.edu/stat/da
p <- within(p, {
  prog <- factor(prog, levels=1:3, labels=c("(
  id <- factor(id)
})
summary(p)
```

Poisson Regression with R

```
       id        num_awards             prog    
 1      :  1   Min.   :0.00   General   : 45  
 2      :  1   1st Qu.:0.00   Academic  :105  
 3      :  1   Median :0.00   Vocational: 50  
 4      :  1   Mean   :0.63                   
 5      :  1   3rd Qu.:1.00                   
 6      :  1   Max.   :6.00                   
 (Other):194                                  
```

Poisson Regression with R

Each variable has 200 valid observations and their distributions seem quite reasonable. The unconditional mean and variance of our outcome variable are not extremely different. Our model assumes that these values, conditioned on the predictor variables, will be equal (or at least roughly so).

Poisson Regression with R

We can use the tapply function to display the summary statistics by program type. The table below shows the average numbers of awards by program type and seems to suggest that program type is a good candidate for predicting the number of awards, our outcome variable, because the mean value of the outcome appears to vary by prog. Additionally, the means and variances within each level of prog–the conditional means and variances–are similar. A conditional histogram separated out by program type is plotted to show the distribution.

Poisson Regression with R

Analysis methods you might consider

- Below is a list of some analysis methods you may have encountered. Some of the methods listed are quite reasonable, while others have either fallen out of favor or have limitations.
- Poisson regression - Poisson regression is often used for modeling count data. Poisson regression has a number of extensions useful for count models.

- Negative binomial regression - Negative binomial regression can be used for over-dispersed count data, that is when the conditional variance exceeds the conditional mean. It can be considered as a generalization of Poisson regression since it has the same mean structure as Poisson regression and it has an extra parameter to model the over-dispersion. If the conditional distribution of the outcome variable is over-dispersed, the confidence intervals for Negative binomial regression are likely to be narrower as compared to those from a Poisson regression.
- Zero-inflated regression model - Zero-inflated models attempt to account for excess zeros. In other words, two kinds of zeros are thought to exist in the data, "true zeros" and "excess zeros". Zero-inflated models estimate two equations simultaneously, one for the count model and one for the excess zeros.

Poisson Regression with R

- OLS regression - Count outcome variables are sometimes log-transformed and analyzed using OLS regression. Many issues arise with this approach, including loss of data due to undefined values generated by taking the log of zero (which is undefined) and biased estimates.

Poisson Regression with R

Poisson regression
At this point, we are ready to perform our Poisson model analysis using the glm function. We fit the model and store it in the object m1 and get a summary of the model at the same time.

Poisson Regression with R

```
summary(m1 <- glm(num_awards ~ prog + math,


 Call:
 glm(formula = num_awards ~ prog + math, fam

 Deviance Residuals:
    Min       1Q   Median       3Q      Max
 -2.204   -0.844   -0.511    0.256    2.680
```

Poisson Regression with R

```
Coefficients:
               Estimate Std. Error z value
(Intercept)     -5.2471     0.6585   -7.97
progAcademic     1.0839     0.3583    3.03
progVocational   0.3698     0.4411    0.84
math             0.0702     0.0106    6.62
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*'
```

Poisson Regression with R

```
(Dispersion parameter for poisson family ta

    Null deviance: 287.67  on 199  degrees
Residual deviance: 189.45  on 196  degrees
AIC: 373.5

Number of Fisher Scoring iterations: 6
```

Poisson Regression with R

Cameron and Trivedi (2009) recommended using robust standard errors for the parameter estimates to control for mild violation of the distribution assumption that the variance equals the mean. We use R package sandwich below to obtain the robust standard errors and calculated the p-values accordingly. Together with the p-values, we have also calculated the 95

Poisson Regression with R

```
cov.m1 <- vcovHC(m1, type="HC0")
std.err <- sqrt(diag(cov.m1))
r.est <- cbind(Estimate= coef(m1), "Robust S
"Pr(>|z|)" = 2 * pnorm(abs(coef(m1)/std.err)
LL = coef(m1) - 1.96 * std.err,
UL = coef(m1) + 1.96 * std.err)
```

Poisson Regression with R

```
r.est

                 Estimate Robust SE  Pr(>|z|)
 (Intercept)     -5.24712   0.64600 4.567e-16
 progAcademic     1.08386   0.32105 7.355e-04
 progVocational   0.36981   0.40042 3.557e-01
 math             0.07015   0.01044 1.784e-11
```

Poisson Regression with R

Now let's look at the output of function glm more closely.

- The output begins with echoing the function call. The information on deviance residuals is displayed next.
- Deviance residuals are approximately normally distributed if the model is specified correctly.In our example, it shows a little bit of skeweness since median is not quite zero.

- Next come the Poisson regression coefficients for each of the variables along with the standard errors, z-scores, p-values and 95% confidence intervals for the coefficients. The coefficient for math is .07.This means that the expected log count for a one-unit increase in math is .07.

- The indicator variable progAcademic compares between prog = "Academic" and prog = "General", the expected log count for prog = "Academic" increases by about 1.1. The indicator variable prog.Vocational is the expected difference in log count ($\approx .37$) between prog = "Vocational" and the reference group (prog = "General").

- The information on deviance is also provided. We can use the residual deviance to perform a goodness of fit test for the overall model. The residual deviance is the difference between the deviance of the current model and the maximum deviance of the ideal model where the predicted values are identical to the observed.

- Therefore, if the residual difference is small enough, the goodness of fit test will not be significant, indicating that the model fits the data. We conclude that the model fits reasonably well because the goodness-of-fit chi-squared test is not statistically significant.
- If the test had been statistically significant, it would indicate that the data do not fit the model well. In that situation, we may try to determine if there are omitted predictor variables, if our linearity assumption holds and/or if there is an issue of over-dispersion.

Poisson Regression with R

```
with(m1, cbind(res.deviance = deviance, df =
  p = pchisq(deviance, df.residual, lower.ta

     res.deviance  df      p
[1,]        189.4 196 0.6182
```

Poisson Regression with R

We can also test the overall effect of prog by comparing the deviance of the full model with the deviance of the model excluding prog. The two degree-of-freedom chi-square test indicates that prog, taken together, is a statistically significant predictor of num_awards.

Poisson Regression with R

```
 update m1 model dropping prog
m2 <- update(m1, . ~ . - prog)
 test model differences with chi square test
anova(m2, m1, test="Chisq")

 Analysis of Deviance Table

 Model 1: num_awards ~ math
 Model 2: num_awards ~ prog + math
   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
 1       198        204
 2       196        189  2     14.6  0.00069
```

Poisson Regression with R

Sometimes, we might want to present the regression results as incident rate ratios and their standard errors, together with the confidence interval. To compute the standard error for the incident rate ratios, we will use the Delta method. To this end, we make use the function deltamethod implemented in R package **msm**.

Poisson Regression with R

```
s <- deltamethod(list(~ exp(x1), ~ exp(x2),

 exponentiate old estimates dropping the p v
rexp.est <- exp(r.est[, -3])
 replace SEs with estimates for exponentiate
rexp.est[, "Robust SE"] <- s
```

Poisson Regression with R

```
rexp.est

                  Estimate Robust SE       LL
 (Intercept)      0.005263   0.00340 0.001484
 progAcademic     2.956065   0.94904 1.575551
 progVocational 1.447458   0.57959 0.660335
 math             1.072672   0.01119 1.050955
```

Poisson Regression with R

The output above indicates that the incident rate for prog = "Academic" is 2.96 times the incident rate for the reference group (prog = "General"). Likewise, the incident rate for prog = "Vocational" is 1.45 times the incident rate for the reference group holding the other variables at constant. The percent change in the incident rate of num_awards is by 7% for every unit increase in math. For additional information on the various metrics in which the results can be presented, and the interpretation of such, please see Regression Models for Categorical Dependent Variables Using Stata, Second Edition by J. Scott Long and Jeremy Freese (2006).

Poisson Regression with R

Sometimes, we might want to look at the expected marginal means. For example, what are the expected counts for each program type holding math score at its overall mean? To answer this question, we can make use of the predict function. First off, we will make a small data set to apply the predict function to it.

Poisson Regression with R

```
(s1 <- data.frame(math = mean(p$math),
  prog = factor(1:3, levels = 1:3, labels =

   math       prog
1 52.65    General
2 52.65   Academic
3 52.65 Vocational
```

Poisson Regression with R

```
predict(m1, s1, type="response", se.fit=TRUE

 $fit
      1      2      3
 0.2114 0.6249 0.3060

 $se.fit
       1       2       3
 0.07050 0.08628 0.08834

 $residual.scale
 [1] 1
```

Poisson Regression with R

In the output above, we see that the predicted number of events for level 1 of prog is about .21, holding math at its mean. The predicted number of events for level 2 of prog is higher at .62, and the predicted number of events for level 3 of prog is about .31. The ratios of these predicted counts ($\frac{.625}{.211} = 2.96$, $\frac{.306}{.211} = 1.45$) match what we saw looking at the IRR.

Poisson Regression with R

We can also graph the predicted number of events with the commands below. The graph indicates that the most awards are predicted for those in the academic program (prog $= 2$), especially if the student has a high math score. The lowest number of predicted awards is for those students in the general program (prog $= 1$). The graph overlays the lines of expected values onto the actual points, although a small amount of random noise was added vertically to lessen overplotting.

Poisson Regression with R

calculate and store predicted values

```
p$phat <- predict(m1, type="response")

 order by program and then by math
p <- p[with(p, order(prog, math)), ]
```

Poisson Regression with R

create the plot

```
ggplot(p, aes(x = math, y = phat, colour = p
  geom_point(aes(y = num_awards), alpha=.5, 
  geom_line(size = 1) +
  labs(x = "Math Score", y = "Expected numbe
```

Poisson Regression with R

Things to consider

- When there seems to be an issue of dispersion, we should first check if our model is appropriately specified, such as omitted variables and functional forms. For example, if we omitted the predictor variable prog in the example above, our model would seem to have a problem with over-dispersion. In other words, a misspecified model could present a symptom like an over-dispersion problem.

Poisson Regression with R

- Assuming that the model is correctly specified, the assumption that the conditional variance is equal to the conditional mean should be checked. There are several tests including the likelihood ratio test of over-dispersion parameter alpha by running the same model using negative binomial distribution. R package pscl (Political Science Computational Laboratory, Stanford University) provides many functions for binomial and count data including odTest for testing over-dispersion.

Poisson Regression with R

- One common cause of over-dispersion is excess zeros, which in turn are generated by an additional data generating process. In this situation, zero-inflated model should be considered.
- If the data generating process does not allow for any 0s (such as the number of days spent in the hospital), then a zero-truncated model may be more appropriate.

Poisson Regression with R

- Count data often have an exposure variable, which indicates the number of times the event could have happened. This variable should be incorporated into a Poisson model with the use of the offset option.
- The outcome variable in a Poisson regression cannot have negative numbers, and the exposure cannot have 0s.

Poisson Regression with R

- Many different measures of pseudo-R-squared exist. They all attempt to provide information similar to that provided by R-squared in OLS regression, even though none of them can be interpreted exactly as R-squared in OLS regression is interpreted.
- Poisson regression is estimated via maximum likelihood estimation. It usually requires a large sample size.

Negative Binomial Regression with R

Introduction Negative binomial regression is for modeling count variables, usually for over-dispersed count outcome variables.

Negative Binomial Regression with R

This page uses the following packages. Make sure that you can load them before trying to run the examples on this page. If you do not have a package installed, run: install.packages("packagename"), or if you see the version is out of date, run: update.packages().

```
require(foreign)
require(ggplot2)
require(MASS)
Version info: Code for this page was tested
On: 2013-01-22
With: MASS 7.3-22; ggplot2 0.9.3; foreign 0
```

Negative Binomial Regression with R

Examples of negative binomial regression

- Example 1. School administrators study the attendance behavior of high school juniors at two schools. Predictors of the number of days of absence include the type of program in which the student is enrolled and a standardized test in math.
- Example 2. A health-related researcher is studying the number of hospital visits in past 12 months by senior citizens in a community based on the characteristics of the individuals and the types of health plans under which each one is covered.

Negative Binomial Regression with R

Description of the data Let's pursue Example 1 from above.

- We have attendance data on 314 high school juniors from two urban high schools in the file **nb_data**.
- The response variable of interest is days absent, daysabs. The variable math gives the standardized math score for each student.
- The variable prog is a three-level nominal variable indicating the type of instructional program in which the student is enrolled.

Negative Binomial Regression with R

Exploratory Data Analysis

```
dat <- read.dta("http://www.ats.ucla.edu/sta
dat <- within(dat, {
prog <- factor(prog, levels = 1:3, labels =
id <- factor(id)
})
```

Negative Binomial Regression with R

```
summary(dat)
         id             gender             math
   1001    :  1    female:160    Min.    : 1.0
   1002    :  1    male  :154    1st Qu.:28.0
   1003    :  1                  Median :48.0
   1004    :  1                  Mean   :48.3
   1005    :  1                  3rd Qu.:70.0
   1006    :  1                  Max.   :99.0
   (Other):308
          prog
   General    : 40
   Academic   :167
```

Negative Binomial Regression with R

```
ggplot(dat, aes(daysabs, fill = prog)) + geom_
., margins = TRUE, scales = "free")
```

Histogram plots showing distribution of the data
Each variable has 314 valid observations and their distributions seem quite reasonable. The unconditional mean of our outcome variable is much lower than its variance.

Negative Binomial Regression with R

Data Set

- Let's continue with our description of the variables in this dataset.
- The table below shows the average numbers of days absent by program type and seems to suggest that program type is a good candidate for predicting the number of days absent, our outcome variable, because the mean value of the outcome appears to vary by prog.
- The variances within each level of prog are higher than the means within each level.
- These are the conditional means and variances. These differences suggest that over-dispersion is present and that a Negative Binomial model

Negative Binomial Regression with R

```
with(dat, tapply(daysabs, prog, function(x) {
sprintf("M (SD) = %1.2f (%1.2f)", mean(x), sd(x))
}))
##                   General                  Academic
## "M (SD) = 10.65 (8.20)"  "M (SD) = 6.93 (7.45)"  "M
```

Analysis methods you might consider

- Below is a list of some analysis methods you may have encountered.
- Some of the methods listed are quite reasonable, while others have either fallen out of favor or have limitations.

Negative Binomial Regression

- Negative binomial regression can be used for over-dispersed count data, that is when the conditional variance exceeds the conditional mean.
- It can be considered as a generalization of Poisson regression since it has the same mean structure as Poisson regression and it has an extra parameter to model the over-dispersion.
- If the conditional distribution of the outcome variable is over-dispersed, the confidence intervals for the Negative binomial regression are likely to be narrower as compared to those from a Poisson regression model

Negative Binomial Regression with R

- Poisson regression - Poisson regression is often used for modeling count data. Poisson regression has a number of extensions useful for count models.
- Zero-inflated regression model - Zero-inflated models attempt to account for excess zeros. In other words, two kinds of zeros are thought to exist in the data, "true zeros" and "excess zeros". Zero-inflated models estimate two equations simultaneously, one for the count model and one for the excess zeros.

Negative Binomial Regression with R

- OLS regression - Count outcome variables are sometimes log-transformed and analyzed using OLS regression. Many issues arise with this approach, including loss of data due to undefined values generated by taking the log of zero (which is undefined), as well as the lack of capacity to model the dispersion.

Negative Binomial Regression with R

Negative binomial regression analysis We will use the `glm.nb` function from the MASS package to estimate a negative binomial regression.

```
summary(m1 <- glm.nb(daysabs ~ math + prog,
##
## Call:
## glm.nb(formula = daysabs ~ math + prog, d
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.155   -1.019   -0.369    0.229    2.527
```

Negative Binomial Regression with R

```
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|
## (Intercept)      2.61527    0.19746   13.24  < 2e-1
## math            -0.00599    0.00251   -2.39    0.01
## progAcademic    -0.44076    0.18261   -2.41    0.01
## progVocational -1.27865    0.20072   -6.37  1.9e-1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '
```

Negative Binomial Regression with R

```
## (Dispersion parameter for Negative Binomi
##
##     Null deviance: 427.54  on 313  degree
## Residual deviance: 358.52  on 310  degree
## AIC: 1741
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  1.033
##           Std. Err.:  0.106
##
```

Negative Binomial Regression with R

- R first displays the call and the deviance residuals.
- Next, we see the regression coefficients for each of the variables, along with standard errors, z-scores, and p-values.
- The variable math has a coefficient of -0.006, which is statistically significant.
- This means that for each one-unit increase in math, the expected log count of the number of days absent decreases by 0.006. The indicator variable shown as progAcademic is the expected difference in log count between group 2 and the reference group (prog=1).

Negative Binomial Regression with R

- The expected log count for level 2 of prog is 0.44 lower than the expected log count for level 1.
- The indicator variable for progVocational is the expected difference in log count between group 3 and the reference group.The expected log count for level 3 of prog is 1.28 lower than the expected log count for level 1. To determine if prog itself, overall, is statistically significant, we can compare a model with and without prog. The reason it is important to fit separate models, is that unless we do, the overdispersion parameter is held constant.

Negative Binomial Regression with R

```
m2 <- update(m1, . ~ . - prog)
anova(m1, m2)
## Likelihood ratio tests of Negative Binomial Models
##
## Response: daysabs
##          Model  theta Resid. df    2 x log-lik.   T
## 1         math 0.8559       312           -1776
## 2 math + prog 1.0327       310           -1731 1 v
##     Pr(Chi)
## 1
## 2 1.652e-10
```

Negative Binomial Regression with R

- The two degree-of-freedom chi-square test indicates that prog is a statistically significant predictor of daysabs.
- The null deviance is calculated from an intercept-only model with 313 degrees of freedom. Then we see the residual deviance, the deviance from the full model. We are also shown the AIC and 2*log likelihood.

Negative Binomial Regression with R

- The theta parameter shown is the dispersion parameter.
- Note that R parameterizes this differently from SAS, Stata, and SPSS.
- The R parameter (theta) is equal to the inverse of the dispersion parameter (alpha) estimated in these other software packages. Thus, the theta value of 1.033 seen here is equivalent to the 0.968 value seen in the Stata Negative Binomial Data Analysis Example because $1/0.968 = 1.033$.

Negative Binomial Regression with R

Checking model assumption

- As we mentioned earlier, negative binomial models assume the conditional means are not equal to the conditional variances.
- This inequality is captured by estimating a dispersion parameter (not shown in the output) that is held constant in a Poisson model.
- Thus, the Poisson model is actually nested in the negative binomial model.
- We can then use a likelihood ratio test to compare these two and test this model assumption.
- To do this, we will run our model as a Poisson.

Negative Binomial Regression with R

```
m3 <- glm(daysabs ~ math + prog, family = "poisson", da
pchisq(2 * (logLik(m1) - logLik(m3)), df = 1, lower.ta
## 'log Lik.' 2.157e-203 (df=5)
```

In this example the associated chi-squared value is 926.03 with one degree of freedom. This strongly suggests the negative binomial model, estimating the dispersion parameter, is more appropriate than the Poisson model.

Negative Binomial Regression with R

We can get the confidence intervals for the coefficients by profiling the likelihood function.

```
(est <- cbind(Estimate = coef(m1), confint(m1
## Waiting for profiling to be done...
##                  Estimate   2.5 %    97.5 %
## (Intercept)      2.615265  2.2421  3.012936
## math            -0.005993 -0.0109 -0.001067
## progAcademic    -0.440760 -0.8101 -0.092643
## progVocational -1.278651 -1.6835 -0.890078
```

Negative Binomial Regression with R

We might be interested in looking at incident rate ratios rather than coefficients. To do this, we can exponentiate our model coefficients. The same applies to the confidence intervals.

Negative Binomial Regression with R

```
exp(est)
##                  Estimate  2.5 %  97.5 %
## (Intercept)       13.6708 9.4127 20.3470
## math               0.9940 0.9892  0.9989
## progAcademic       0.6435 0.4448  0.9115
## progVocational     0.2784 0.1857  0.4106
```

Negative Binomial Regression with R

The output above indicates that the incident rate for prog $= 2$ is 0.64 times the incident rate for the reference group (prog $= 1$). Likewise, the incident rate for prog $= 3$ is 0.28 times the incident rate for the reference group holding the other variables constant. The percent change in the incident rate of daysabs is a 1

Negative Binomial Regression with R

The form of the model equation for negative binomial regression is the same as that for Poisson regression. The log of the outcome is predicted with a linear combination of the predictors:

Negative Binomial Regression with R

$$ln(\widehat{daysabs_i}) = Intercept + b_1(prog_i = 2) + b_2(prog_i = 3) +$$

$$\therefore$$

$$\widehat{daysabs_i} = e^{Intercept + b_1(prog_i=2) + b_2(prog_i=3) + b_3 math_i} = e^{Intercept}$$

The coefficients have an additive effect in the $ln(y)$ scale and the IRR have a multiplicative effect in the y scale. The dispersion parameter in negative binomial regression does not effect the expected counts, but it does effect the estimated variance of the expected counts.

Negative Binomial Regression with R

- More details can be found in the Modern Applied Statistics with S by W.N. Venables and B.D. Ripley (the book companion of the MASS package).
- For additional information on the various metrics in which the results can be presented, and the interpretation of such, please see Regression Models for Categorical Dependent Variables Using Stata, Second Edition by J. Scott Long and Jeremy Freese (2006).

Negative Binomial Regression with R

Predicted values

- For assistance in further understanding the model, we can look at predicted counts for various levels of our predictors. Below we create new datasets with values of math and prog and then use the predict command to calculate the predicted number of events.

Negative Binomial Regression with R

- First, we can look at predicted counts for each value of prog while holding math at its mean. To do this, we create a new dataset with the combinations of prog and math for which we would like to find predicted values, then use the predict command.

Negative Binomial Regression with R

```
newdata1 <- data.frame(math = mean(dat$math)
labels = levels(dat$prog)))
newdata1$phat <- predict(m1, newdata1, type
newdata1
   math       prog   phat
 1 48.27    General 10.237
 2 48.27   Academic  6.588
 3 48.27 Vocational  2.850
```

Negative Binomial Regression with R

- In the output above, we see that the predicted number of events (e.g., days absent) for a general program is about 10.24, holding math at its mean.
- The predicted number of events for an academic program is lower at 6.59, and the predicted number of events for a vocational program is about 2.85.

Negative Binomial Regression with R

Below we will obtain the mean predicted number of events for values of math across its entire range for each level of prog and graph these.

Negative Binomial Regression with R

```
newdata2 <- data.frame(
math = rep(seq(from = min(dat$math),
       to = max(dat$math), length.out = 100)
prog = factor(rep(1:3, each = 100), levels =
levels(dat$prog)))
```

Negative Binomial Regression with R

```
newdata2 <- cbind(newdata2, predict(m1, newdat
newdata2 <- within(newdata2, {
DaysAbsent <- exp(fit)
LL <- exp(fit - 1.96 * se.fit)
UL <- exp(fit + 1.96 * se.fit)
})
```

Negative Binomial Regression with R

```
ggplot(newdata2, aes(math, DaysAbsent)) +
geom_ribbon(aes(ymin = LL, ymax = UL, fill = prog), a
geom_line(aes(colour = prog), size = 2) +
labs(x = "Math Score", y = "Predicted Days Absent")
```

Negative Binomial Regression with R

Plot of the model predicted days absent with confidence intervals The graph shows the expected count across the range of math scores, for each type of program along with 95 percent confidence intervals. Note that the lines are not straight because this is a log linear model, and what is plotted are the expected values, not the log of the expected values.

Negative Binomial Regression with R

Things to consider

- It is not recommended that negative binomial models be applied to small samples.
- One common cause of over-dispersion is excess zeros by an additional data generating process.
- In this situation, zero-inflated model should be considered.

Negative Binomial Regression with R

If the data generating process does not allow for any 0s (such as the number of days spent in the hospital), then a zero-truncated model may be more appropriate.

Negative Binomial Regression with R

Count data often have an exposure variable, which indicates the number of times the event could have happened. This variable should be incorporated into your negative binomial regression model with the use of the offset option. See the glm documentation for details.

Negative Binomial Regression with R

The outcome variable in a negative binomial regression cannot have negative numbers. You will need to use the `m1$resid` command to obtain the residuals from our model to check other assumptions of the negative binomial model (see Cameron and Trivedi (1998) and Dupont (2002) for more information).

Multinomial Logistic Regression with R

```
## melt data set to long for ggplot2
lpp <- melt(pp.write, id.vars = c("ses", "write"), value.
head(lpp)  # view first few rows

##    ses write variable probability
## 1 low     30 academic     0.09844
## 2 low     31 academic     0.10717
## 3 low     32 academic     0.11650
## 4 low     33 academic     0.12646
## 5 low     34 academic     0.13705
## 6 low     35 academic     0.14828
```

Multinomial Logistic Regression with R

```
## plot predicted probabilities across write values for e
## facetted by program type
ggplot(lpp, aes(x = write, y = probability, colour = ses)
    ., scales = "free")
```

Ordinal Logistic Regression with R

```
ggplot(dat, aes(x = apply, y = gpa)) +
  geom_boxplot(size = .75) +
  geom_jitter(alpha = .5) +
  facet_grid(pared ~ public, margins = TRUE) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1,
```

Poisson Regression with R

```
with(p, tapply(num_awards, prog, function(x) {
  sprintf("M (SD) = %1.2f (%1.2f)", mean(x), sd(x))
}))
```

Poisson Regression with R

```
##                         General                         Academic
## "M (SD) = 0.20 (0.40)" "M (SD) = 1.00 (1.28)" "M (SD)

ggplot(p, aes(num_awards, fill = prog)) +
  geom_histogram(binwidth=.5, position="dodge")
```

Negative Binomial Regression with R

```
ggplot(dat, aes(daysabs, fill = prog)) + geom_histogram(b
    ., margins = TRUE, scales = "free")
```

```
with(dat, tapply(daysabs, prog, function(x) {
    sprintf("M (SD) = %1.2f (%1.2f)", mean(x), sd(x))
}))
```

```
##                  General                  Academic
## "M (SD) = 10.65 (8.20)"  "M (SD) = 6.93 (7.45)"  "M (S
```