R Data Analysis Examples
Ordinal Logistic Regression

Note: the code on this page works with R 2.4.1 and requires the library Design.

Examples

Example 1: A marketing research firm wants to investigate what factors influence the size of soda (small, medium, large or extra large) that people order at a fast-food chain. These factors may include what type of sandwich is ordered (burger or chicken), whether or not fries are also ordered, and age of the consumer. While the outcome variable, size of soda, is obviously ordered, the difference between the various sizes is not consistent. The differences are 10, 8, 12 ounces, respectively.

Example 2: A 5-point Likert scale is used to assess people's opinion about a local ballot measure. The response options are "strongly disagree", "disagree", "neutral", "agree" and "strongly agree". Predictor variables will include the measure's author, his/her political party, and how much the measure's proposals will cost. The researchers have reason to believe that the psychological "distances" between these points are not equal. For example, the "distance" between "strongly disagree" and "disagree" may be shorter than the distance between "disagree" and "neutral".

Example 3: A study looks at factors that influence the decision of whether to apply to graduate school. College juniors are asked if they are unlikely, somewhat likely, or very likely to apply to graduate school. Hence, our outcome variable has three categories. Data on parental educational status, whether the undergraduate institution is public or private, and current GPA is also collected.

Description of the Data

For our data analysis below, we are going to expand on Example 3 about applying to graduate school. We have generated hypothetical data, which can be obtained from our website using the following code:

```
mydata <- read.csv(url("http://www.ats.ucla.edu/stat/r/dae/ologit.csv"))
attach(mydata)
names(mydata)
[1] "apply"  "pared"  "public" "gpa"
```

This hypothetical data set has a three level variable called apply (coded 0, 1, 2), that we will use as our response (i.e., outcome, dependent) variable. We also have three variables that we will use as predictors: pared, which is a 0/1 variable indicating whether at least one parent has a graduate degree; public, which is a 0/1 variable where 1 indicates that the undergraduate institution is a public university and 0 indicates that it is a private university, and gpa, which is the student's grade point average.

```
table(apply)
apply
0   1   2
220 140  40

table(pared)
pared
  0   1
337  63

table(public)
```

```
public
  0   1
343  57

summary(gpa)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
1.900   2.720   2.990   2.999   3.270   4.000

sd(gpa)
[1] 0.3979409
```

Some Strategies You Might Try

- OLS regression: This analysis is problematic because the assumptions of OLS are violated when it is used with a non-interval outcome variable.
- ANOVA: If you use only one continuous predictor, you could "flip" the model around so that, say, gpa was the outcome variable and apply was the predictor variable. Then you could run a one-way ANOVA. This isn't a bad thing to do if you only have one predictor variable (from the logistic model), and it is continuous.
- Multinomial logistic regression: This is similar to doing ordinal logistic regression, except that it is assumed that there is no order to the categories of the outcome variable (i.e., the categories are nominal). The downside of this approach is that the information contained in the ordering is lost.
- Ordinal probit regression: This is very, very similar to running an ordinal logistic regression. The main difference is in the interpretation of the coefficients.

Using the Ordinal Logistic Model

Before we run our ordinal logistic model, we will see if any cells (created by the crosstab of our categorical and response variables) are empty or extremely small. If any are, we may have difficulty running our model.

```
xtabs(~ pared + apply)
     apply
pared   0   1   2
   0 200 110  27
   1  20  30  13

xtabs(~ public + apply)
      apply
public   0   1   2
     0 189 124  30
     1  31  16  10
```

None of the cells is too small or empty (have no cases), so we will run our model. In order to run ordinal logistic regression in R, you first need to install the package (also called a library) Design. Once you have installed Design, you will need to tell R you want to use this library before you can access the commands in the Design library. This is called "loading" the library. You load the library using the command library( ) shown in the first line of code below. The next two lines of code store certain properties of the variables listed and their distributions, this information can later be used by the Design package to show the effects of the predictor variables in the model. The first line gets this information and stores it in ddist, while the second sets ddist as the data distribution to be used in the future. The following (fourth) line of code specifies the model. lrm( ) is the R function we will use to model the data. Within the parentheses first we see the model apply~pared+public+gpa. The command data=mydata sets the data frame to be analyzed. Finally, na.action=na.pass, tells R that when it encounters a missing value, it should omit that case. The final line of code asks for a summary of the model, which is shown below.

```
library(Design)
ddist<- datadist(pared, public, gpa)
```

```
options(datadist='ddist')
ologit<- lrm(apply ~ pared + public + gpa, data=mydata, na.action=na.pass)
print(ologit)

Logistic Regression Model

lrm(formula=apply~pared+public+gpa, data = mydata, na.action = na.pass)

Frequencies of Responses
  0   1   2
220 140  40

      Obs  Max Deriv Model L.R.       d.f.          P          C        Dxy
      400       3e-11       24.18          3          0      0.605       0.21
    Gamma      Tau-a         R2      Brier
    0.212      0.119       0.07      0.235

      Coef     S.E.   Wald Z P
y>=1   -2.20332 0.7795 -2.83  0.0047
y>=2   -4.29877 0.8043 -5.34  0.0000
pared   1.04766 0.2658  3.94  0.0001
public -0.05868 0.2979 -0.20  0.8438
gpa     0.61575 0.2606  2.36  0.0182
```

In the output above, first we see a label (Logistic Regression Model) followed by the call, this is R reminding us what type of model we ran, what options we specified, etc.. Next we see the frequency for each level of the dependent variable. Next we see additional information on the model. The first column of the first row (labeled obs) gives the number of observations included in the analysis (400). Moving over two columns, we see the chi-square likelihood ratio (labeled L.R.), followed by the degrees of freedom for the model (3) and the approximate p-value for the model (0). The likelihood ratio chi-square of 24.18 with three degrees of freedom and a p-value of approximately 0, tells us that our model as a whole is statistically significant, as compared to model with no predictors. In the second row, the column labeled R2 gives a pseudo-R-squared for the model. It is a pseudo-R-squared because there is no direct equivalent of an R-squared (from OLS regression) in non-linear models. There are many different pseudo-R-squares, but the emphasis should be on the pseudo.

Next we see the table of coefficients, their standard errors, the (Wald) z-test, and associated p-values. The first two coefficients in the model are the cut-points which indicate where the latent variable is cut to make the three groups we observe in our data. Note that this latent variable is continuous. In general, these are not used in the interpretation of the results. The cut-points are closely related to thresholds, which are reported by some other statistical packages. The rest of the table shows the coefficients for our predictor variables. Both pared and gpa are statistically significant; public is not. 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. 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.

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. The Design package can output these as well. The summary command (used below) requests a summary of our model (ologit). The Design package produces a summary table that looks different from some other summary tables produced by R (e.g. a summary table of coefficients for a binary logistic regression). The table contains two rows for each of the independent variables in our model, the first, labeled with the name of the variable, gives the effects in the logit scale, the second labeled Odds Ratio gives the odds ratio for a given change in the predictor variable. For dummy variables, the columns labeled Low and High will always be 0 and 1 respectively, and the Diff. column will always be 1. For continuous variables, the Low column is the first quartile, and the High column is the third quartile, and the Diff. column is their difference (High-Low). The column labeled Effect contains the coefficients for a change in

the x-variable from the value in the Low column to the value in the High column. 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 high apply versus the combined middle and low categories are 2.85 greater, given that all of the other variables in the model are held constant. Likewise, the odds of the combined middle and high categories versus low apply 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 from 2.72 (Low or first quartile) to 3.27 (High or third quartile) the odds of moving from the low to middle or high categories of apply (or from the lower and middle categories to the high category) is multiplied by 1.40. The rest of the columns contain the standard error of the effect, as well as the lower and upper 95% confidence intervals.

```
summary(ologit)

          Effects          Response : apply

  Factor      Low  High Diff. Effect S.E. Lower 0.95 Upper 0.95
  pared      0.00 1.00 1.00   1.05  0.27  0.53       1.57
   Odds Ratio 0.00 1.00 1.00   2.85    NA  1.69       4.80
  public     0.00 1.00 1.00  -0.06  0.30 -0.64       0.53
   Odds Ratio 0.00 1.00 1.00   0.94    NA  0.53       1.69
  gpa        2.72 3.27 0.55   0.34  0.14  0.06       0.62
   Odds Ratio 2.72 3.27 0.55   1.40    NA  1.06       1.86
```

But what if we wanted to know the change in odds of moving from one value of apply to the next for some other change in the predictor variables? (For example, a one unit change in gpa.) We can do this as well. In the code below we have added gpa=c(3,4) which indicates that we want to get the same information about the coefficients, but for gpa we want the Low value to be 3, and the High value to be 4. Note that since the effect of a one unit change in x is constant across values of that variable, we could have used any one unit change in gpa (e.g. c(2,3) or c(2.5,3.5)) and gotten exactly the same results. We can specify the values we wish to use for any number of the predictor variables in our model, for example, if we had another variable in our model called hours, for number of hours spent studying per week, and we wanted to see the change in odds ratio as it changed from 10 to 20, as well as for when gpa changed from 2 to 3, we would use the command summary(ologit, gpa=c(3,4), hours=c(10,20)).

```
summary(ologit, gpa=c(3,4))

          Effects          Response : apply

  Factor      Low High Diff. Effect S.E. Lower 0.95 Upper 0.95
  pared       0   1    1      1.05  0.27  0.53       1.57
   Odds Ratio 0   1    1      2.85    NA  1.69       4.80
  public      0   1    1     -0.06  0.30 -0.64       0.53
   Odds Ratio 0   1    1      0.94    NA  0.53       1.69
  gpa         3   4    1      0.62  0.26  0.10       1.13
   Odds Ratio 3   4    1      1.85    NA  1.11       3.09
```

In this table we see that for a one unit increase in gpa, the odds of the low and middle categories of apply versus the high category of apply are 1.85 times greater, given that the other variables in the model are held constant. Because of the proportional odds assumption (see below for more explanation), the same increase, 1.85 times, is found between low apply and the combined categories of middle and high apply.

Testing the proportional odds assumption

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. 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. Statistical tests to do this are available in some software packages. However, these tests have been criticized for having a tendency to reject this 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. 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. Fortunately, the Hmisc library is loaded automatically when the Design library is loaded, so you probably don't need to worry about this. The code below contains three commands (the first command falls on multiple lines). The first command creates the function that estimates the values that will be graphed. The first line of this command tells R that sf gets a function, and that this function takes one argument, which we label y. The following two lines contain the code you want the function to execute. 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, 0,1, 2. If your dependent variable has 4 levels, labeled 0, 1, 2, 3 you would need to add ", 'Y>=3'=qlogis(mean(y >= 3))" (minus the quotation marks) inside the first set of parentheses. If your dependent variable were coded 1, 2 ,3 instead of 0, 1, 2, you would need to edit the code, replacing each instance of 0 with 1, 1 with 2, and so on. The second command below instructs R to create an object called s, and that it should contain a summary( ) based on the formula apply ~ pared + public + gpa (note this is the formula used for our model), and fun=sf tells R that before it gives us a summary, we want to apply the function sf to the formula. The final command asks R to return the contents of the object s, which is a table.

```
sf <- function(y)
    c('Y>=0'=qlogis(mean(y >= 0)),'Y>=1'=qlogis(mean(y >= 1)),
    'Y>=2'=qlogis(mean(y >= 2)))

s <- summary(apply ~ pared + public + gpa, fun=sf)
s

apply    N=400

+------+----------+---+----+----------+--------+
|      |          |N  |Y>=0|Y>=1      |Y>=2    |
+------+----------+---+----+----------+--------+
|pared |No        |337|Inf |-0.37833644|-2.440735|
|      |Yes       | 63|Inf | 0.76546784|-1.347074|
+------+----------+---+----+----------+--------+
|public|No        |343|Inf |-0.20479441|-2.345006|
|      |Yes       | 57|Inf |-0.17589067|-1.547563|
+------+----------+---+----+----------+--------+
|gpa   |[1.90,2.73)|102|Inf |-0.39730180|-2.772589|
|      |[2.73,3.00)| 99|Inf |-0.26415158|-2.302585|
|      |[3.00,3.28)|100|Inf |-0.20067070|-2.090741|
|      |[3.28,4.00]| 99|Inf | 0.06062462|-1.803594|
+------+----------+---+----+----------+--------+
|Overall|          |400|Inf |-0.20067070|-2.197225|
+------+----------+---+----+----------+--------+
```

The table above displays the (linear) predicted values we would get if we regressed our dependent variable on our predictor variables one at a time, without the parallel slopes assumption. The parallel slopes assumption is relaxed by running a series of binary logistic regressions, in which the binary dependent variable is equal to zero if the ordinal variable is less than some value a, and 1 if the ordinal variable is greater than or equal to a (note, this

is what the ordinal regression model coefficients represent as well). This is done for k-1 levels of the ordinal variable. Then, each predictor is regressed on each of these dummy variables. The code below reproduces the coefficients seen in the row of the table labeled "pared." The first line of the code creates a new variable called napply1 that is equal to one if apply is greater than or equal to 1, and equal to zero otherwise. The second line of code runs a binary logit model where napply1 is the dependent variable, and pared is the only predictor variable. Looking at the intercept for this model (-0.3783), we see that it matches the predicted value in the cell for pared equal to "no" in the column for Y>=1, the value below it, for pared equals "yes" is equal to the intercept plus the coefficient for pared (i.e. -0.3783 + 1.1438 = 0.765).

```
mydata$napply1<-as.factor(mydata$apply>=1)
glm(mydata$napply1~mydata$pared, family=binomial(link="logit"))

Call:  glm(formula = mydata$napply1 ~ mydata$pared, family = binomial(link = "logit"))

Coefficients:
 (Intercept)  mydata$pared
     -0.3783        1.1438

Degrees of Freedom: 399 Total (i.e. Null);  398 Residual
Null Deviance:      550.5
Residual Deviance: 534.1        AIC: 538.1
```

These lines of code reproduce the predicted values for apply greater than or equal to 2.

```
mydata$napply1<-as.factor(mydata$apply>=2)
glm(mydata$napply1~mydata$pared, family=binomial(link="logit"))

Call:  glm(formula = mydata$napply1 ~ mydata$pared, family = binomial(link = "logit"))

Coefficients:
 (Intercept)  mydata$pared
      -2.441         1.094

Degrees of Freedom: 399 Total (i.e. Null);  398 Residual
Null Deviance:      260.1
Residual Deviance: 252.2        AIC: 256.2
```

We can use the values in this table to help us assess whether the proportional odds assumption is reasonable for our model. (Note, the table is reproduced below, as well as above.) For example, when pared is equal to "no" the difference between the predicted value for apply greater than or equal to one and apply greater than or equal to two is roughly 2 (-0.378 - -2.440 = 2.062). For pared equal to "yes" the difference in predicted values for apply greater than or equal to one and apply greater than or equal to 2 is similar (0.765 - -1.347 = 2.112). This suggests that the parallel slopes assumption is reasonable (these differences is what is focused on in the graph below). 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 one, versus apply greater than or equal to two 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). 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."

The other reason for producing this table is that it allows us to find the highest and lowest coefficient values (which are highlighted below), these values must be specified in the code for the graph below.

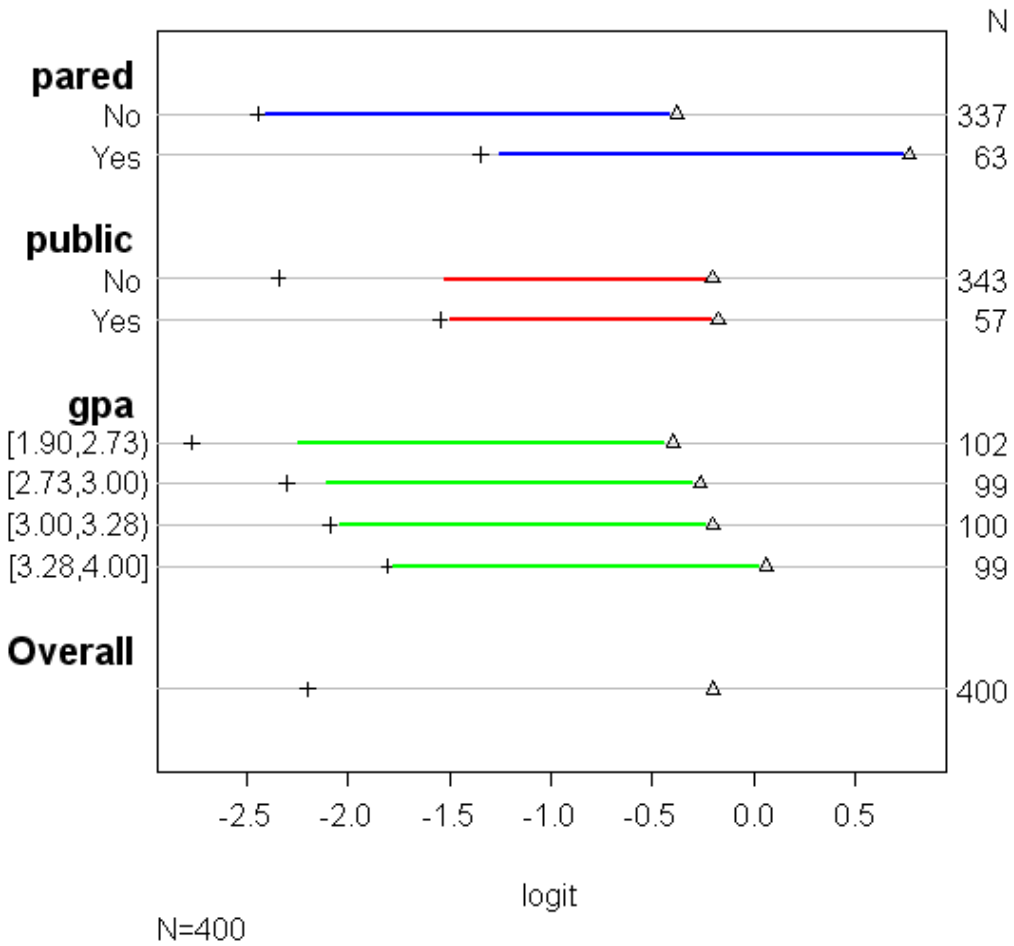
```
apply    N=400

+-------+----------+---+----+----------+---------+
```

```
|      |           |N  |Y>=0|Y>=1       |Y>=2     |
+------+-----------+---+----+-----------+---------+
|pared  |No         |337|Inf |-0.37833644|-2.440735|
|      |Yes        | 63|Inf | 0.76546784|-1.347074|
+------+-----------+---+----+-----------+---------+
|public |No         |343|Inf |-0.20479441|-2.345006|
|      |Yes        | 57|Inf |-0.17589067|-1.547563|
+------+-----------+---+----+-----------+---------+
|gpa    |[1.90,2.73)|102|Inf |-0.39730180|-2.772589|
|      |[2.73,3.00)| 99|Inf |-0.26415158|-2.302585|
|      |[3.00,3.28)|100|Inf |-0.20067070|-2.090741|
|      |[3.28,4.00]| 99|Inf | 0.06062462|-1.803594|
+------+-----------+---+----+-----------+---------+
|Overall|           |400|Inf |-0.20067070|-2.197225|
+------+-----------+---+----+-----------+---------+
```

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. Finally, `xlim=c(-2.8,.8)` is necessary as it sets the range of the x-axis. For our example, all of the values to be plotted fall between -2.8 and .8 (see the table above), if this was not the case, we would not be able to see those points that fall outside this range, because the plot would not extend far enough in either direction. (Note, the blue, red, and green lines are not part of the graph as produced by R, they were added externally for illustrative purposes, to see the original graph from R, click here.)

```
plot(s, which=1:3, pch=1:3, xlab='logit', main=' ', xlim=c(-2.8,.8))
```



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 have added colored lines to the graph produced by R. Each set of lines (by color) are the same length. Looking at the coefficients for the variable `pared` we see that the distance between the two sets of coefficients is similar. That is, the blue line fits between both sets of estimates very well. 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.

Obtaining odds ratios and predicted probabilities to help interpret our results.

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 fix `public` and `gpa` at their means and calculate the probability of being in each category of apply when at least one parent had a graduate degree (`pared = 1`) or otherwise. The first line of the code shown below creates an object called `pared`, it is a vector containing the two values that the variable `pared` takes on (0 and 1). The second line of the code creates an object called `public` that takes on only one value, in this case we have set `public` to its mean using the command `mean(data.frame$varname)`, but it could have been set at any reasonable value (e.g. zero, one, or for continuous variables the median, etc.). The third line does the same except that it creates an object called `gpa` and is equal to the mean of `gpa`. *These objects must have the same names as the variables in your ordinal logistic regression above (e.g. in this example the mean for `gpa` must be named `gpa`).* On the third line of code, we combine the objects we just created (pared, public, and gpa) into a data frame called `newdata1`. This is the new dataset we will used to create our predicted probabilities.

```
pared<-c(0,1))
public<-mean(mydata$public)
gpa<-mean(mydata$gpa)
newdata1<-data.frame(pared,public,gpa)
```

Now that we have the data frame we want to use to calculate the predicted probabilities, we can tell R to create the predicted probabilities. While relatively compact, the first line of code below is important and contains a lot of instructions. The newdata1$predicted tells R that we want to create a new variable in the dataset (data frame) newdata1 called predicted, and that the values of predicted should be predictions made using the predict( ) command. The commands within the parentheses tell R that the predictions should be based on the object (model) ologit, with values of the predictor variables coming from newdata1 and that the type of prediction is a predicted probability (type="fitted.ind"). The second line of the code tells R to list the values in the data frame newdata1. Although not particularly pretty, this is a table of predicted probabilities. The values of pared, public, and gpa are listed (note public and gpa are constants), followed by three columns labeled "predicted.V1," "predicted.1," and "predicted.V3," these contain the predicted probability that apply equals 0, 1 and 2 respectively. As you can see, the predicted probability of being in the lowest category of apply is 0.59 if neither parent has a graduate level education and 0.34 otherwise. For the middle category of apply, the predicted probabilities are 0.33 and 0.46, and for the highest category of apply, 0.079 and 0.196. Hence, if neither of a respondent 's parents have a graduate level education, the predicted probability of applying to graduate school decreases.

```
newdata1$predicted<-predict(ologit,newdata=newdata1,type="fitted.ind")
newdata1
  pared public      gpa predicted.V1 predicted.1 predicted.V3
1     0 0.1425 2.998925   0.59027641  0.33105328   0.07867032
2     1 0.1425 2.998925   0.33569097  0.46853000   0.19577903
```

We can do something very similar to create a table of predicted probabilities varying the value of gpa. Above, we created objects called pared, public and gpa, and then bound them into a single data frame. This time, we are going to create newdata2 by creating a data frame and declaring the values within it at the same time. In the first line below, the command data.frame( ) tells R that we want to create a data frame. Within the parentheses, the command pared=mean(mydata$pared) tells R that the first variable in the data frame should be called pared and that it should equal to the mean of pared for every case in the data frame. The command public=mean(mydata$public) does the same for for the variable public. The command gpa=seq(2,4,1) tells R that the third variable in the data frame should be called gpa and that it should take on a sequence of values from 2 to 4, in increments of 1. The second line of code creates a new variable in dataframe2 called predicted, which contains the predicted probability that each case in newdata2 will belong to each of the categories of apply. Finally, the command newdata2 prints the data frame. As before, the predicted probability are printed in a series of columns labeled "predicted.V1" "predicted.1" and "predicted.V3" with the first containing the predicted probability that a case with those attributes will be in the lowest category of apply (apply=0), the second containing the probability for the middle category of apply, and the third containing the predicted probability for the highest category of apply. As you can see, except for students with a gpa of 4.0, the highest predicted probability is for the lowest category of apply, which makes sense because most respondents are in that category. You can also see that the predicted probability increases for both the middle and highest categories of apply as gpa increases.

```
newdata2<-data.frame(pared=mean(mydata$pared),public=mean(mydata$public),gpa=seq(2,4,1))
newdata2$predicted<-predict(ologit,newdata=newdata2,type="fitted.ind")
newdata2

   pared public gpa predicted.V1 predicted.1 predicted.V3
1 0.1575 0.1425   2   0.69321314  0.25515619   0.05163067
2 0.1575 0.1425   3   0.54969515  0.35875721   0.09154764
3 0.1575 0.1425   4   0.39740099  0.44538933   0.15720968
```

We can use a similar set of commands to get a predicted probability for a respondent with specific attributes. This is useful when you want to describe profiles of respondents. The commands to do this are:

```
newdata3<-data.frame(pared=1,1,gpa=3.5)
newdata3$predicted<-predict(ologit,newdata=newdata3,type="fitted.ind")
newdata3
  pared X1 gpa predicted.V1 predicted.1 predicted.V3
1     1  1 3.5    0.2706966   0.4803782    0.2489252
```

Sample Write-up of the Analysis

Using the print function of the Design package, specifying a smaller number of digits (the default is digits=4) produces a table similar to those seen in some publications. Following this table is an example of one way of describing the results.

```
print(ologit, digits=3)

       Coef   S.E. Wald Z P
y>=1   -2.203 0.78 -2.83  0.0047
y>=2   -4.299 0.80 -5.34  0.0000
pared   1.048 0.27  3.94  0.0001
public -0.059 0.30 -0.20  0.8438
gpa     0.616 0.26  2.36  0.0182
```

Parental education and grade point average are positively associated with the tendency to apply for graduate school. For a one unit increase in pared, the expected ordered log odds increases by 1.05 as you move to the next higher category of apply. For every unit increase in gpa, we expect a 0.62 increase in the expected log odds as you move to the next higher category of apply. There was no statistically significant effect of public on apply.

Describing the results in terms of ordered log-odds (or odds ratios) may not be the simplest metric for your audience to understand. As we saw above, you can also obtain predicted probabilities. These are often useful for helping to tell the "story" of your results.

Similarities and differences between logit and probit models

Neither the ordinal logistic model nor the ordinal probit model are linear. To make the model linear, a transformation is done on the dependent variable. In logistic regression (including binary, ordinal and multinomial logistic models), the transformation is the logit function which is the natural log of the odds. In probit models (including binary, ordinal and multinomial probit models), the function used is the inverse of the standard normal cumulative distribution (a.k.a. a z-score). In reality, this difference isn't too important: both transformations are equally good at linearizing the model; which one you use is a matter of personal preference. Both methods use maximum likelihood, and so require more cases than a similar OLS model. Unlike logistic models, you don't get odds ratios with probit models, instead you get z-scores, and you can get predicted probabilities from either type of model.

Cautions, Flies in the Ointment

- Perfect prediction: Perfect prediction means that only one value of a predictor variable is associated with only one value of the response variable. To avoid this, you should check your data carefully before attempting to run your model.
- Sample size: Both ordinal logistic and ordinal probit, using maximum likelihood estimates, require sufficient sample size. How big is big is a topic of some debate, but they almost always require more cases than OLS regression.
- 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 (a small cell), the model may become unstable or it might not run at all.
- Diagnostics: Doing diagnostics for non-linear models is difficult, and ordered logit/probit models are even more difficult than binary models.

References

Frank E. Harrell, Jr. *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression and Survival Analysis.* Springer, New York, 2001. [335-337]

See Also

Interpreting logistic regression in all its forms (in Adobe .pdf form) (from Stata STB53, Courtesy of, and Copyright, Stata Corporation)
Regression Models for Categorical and Limited Dependent Variables by J. Scott Long
An Introduction to Categorical Data Analysis by Alan Agresti
Categorical Data Analysis, Second Edition by Alan Agresti
Interpreting Probability Models: Logit, Probit, and Other Generalized Linear Models by Tim Futing Liao
Statistical Methods for Categorical Data Analysis by Daniel Powers and Yu Xie

---

How to cite this page Report an error on this page
