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.

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 If you do not have a package installed, run: **install.packages("packagename")**, or if you see the version is out of date, run: **update.packages()**.

```
library(aod)
library(ggplot2)
```

Version info: Code for this page was tested in R version 3.0.2 (2013-09-25)
On: 2013-12-16
With: knitr 1.5; ggplot2 0.9.3.1; aod 1.3

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 and potential follow-up analyses.

Examples

Example 1. Suppose that we are interested in the factors that influence whether a political candidate wins an election. The outcome (response) variable is binary (0/1); win or lose. The predictor variables of interest are the amount of money spent on the campaign, the amount of time spent campaigning negatively and whether or not the candidate is an incumbent.

Example 2. A researcher is interested in how variables, such as GRE (Graduate Record Exam scores), GPA (grade point average) and prestige of the undergraduate institution, effect admission into graduate school. The response variable, admit/don't admit, is a binary variable.

Description of the data

For our data analysis below, we are going to expand on Example 2 about getting into graduate school. We have generated hypothetical data, which can be obtained from our website from within R. Note that *R requires forward slashes* (/) not back slashes () when specifying a file

location even if the file is on your hard drive.

```
mydata <- read.csv("https://stats.idre.ucla.edu/stat/data/binary.csv")
## view the first few rows of the data
head(mydata)
```

```
##   admit gre  gpa rank
## 1     0 380 3.61    3
## 2     1 660 3.67    3
## 3     1 800 4.00    1
## 4     1 640 3.19    4
## 5     0 520 2.93    4
## 6     1 760 3.00    2
```

This dataset has a binary response (outcome, dependent) variable called **admit**. There are three predictor variables: **gre**, **gpa** and **rank**. We will treat the variables **gre** and **gpa** as continuous. The variable **rank** takes on the values 1 through 4. Institutions with a rank of 1 have the highest prestige, while those with a rank of 4 have the lowest. We can get basic descriptives for the entire data set by using **summary**. To get the standard deviations, we use **sapply** to apply the **sd** function to each variable in the dataset.

```
summary(mydata)
```

```
##      admit            gre             gpa             rank
##  Min.   :0.000   Min.   :220   Min.   :2.26   Min.   :1.00
##  1st Qu.:0.000   1st Qu.:520   1st Qu.:3.13   1st Qu.:2.00
##  Median :0.000   Median :580   Median :3.40   Median :2.00
##  Mean   :0.318   Mean   :588   Mean   :3.39   Mean   :2.48
##  3rd Qu.:1.000   3rd Qu.:660   3rd Qu.:3.67   3rd Qu.:3.00
##  Max.   :1.000   Max.   :800   Max.   :4.00   Max.   :4.00
```

```
sapply(mydata, sd)
```

```
##   admit     gre     gpa    rank
##   0.466 115.517   0.381   0.944
```

```
## two-way contingency table of categorical outcome and predictors we want
## to make sure there are not 0 cells
xtabs(~admit + rank, data = mydata)
```

```
##        rank
## admit   1  2  3  4
##     0  28 97 93 55
##     1  33 54 28 12
```

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.

- Logistic regression, the focus of this page.
- Probit regression. Probit analysis will produce results similar logistic regression. The choice of probit versus logit depends largely on individual preferences.
- OLS regression. When used with a binary response variable, this model is known as a linear probability model and can be used as a way to describe conditional probabilities. However, the errors (i.e., residuals) from the linear probability model violate the homoskedasticity and normality of errors assumptions of OLS regression, resulting in invalid standard errors and hypothesis tests. For a more thorough discussion of these and other problems with the linear probability model, see Long (1997, p. 38-40).
- Two-group discriminant function analysis. A multivariate method for dichotomous outcome variables.
- Hotelling's T^2. The 0/1 outcome is turned into the grouping variable, and the former predictors are turned into outcome variables. This will produce an overall test of significance but will not give individual coefficients for each variable, and it is unclear the extent to which each "predictor" is adjusted for the impact of the other "predictors."

Using the logit model

The code below estimates a logistic regression model using the **glm** (generalized linear model) function. First, we convert **rank** to a factor to indicate that rank should be treated as a categorical variable.

```
mydata$rank <- factor(mydata$rank)
mylogit <- glm(admit ~ gre + gpa + rank, data = mydata, family = "binomial")
```

Since we gave our model a name (**mylogit**), R will not produce any output from our regression. In order to get the results we use the summary command:

```
summary(mylogit)
```

```
##
## Call:
## glm(formula = admit ~ gre + gpa + rank, family = "binomial",
##      data = mydata)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.627  -0.866  -0.639   1.149   2.079
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.98998    1.13995   -3.50  0.00047 ***
## gre          0.00226    0.00109    2.07  0.03847 *
## gpa          0.80404    0.33182    2.42  0.01539 *
## rank2       -0.67544    0.31649   -2.13  0.03283 *
## rank3       -1.34020    0.34531   -3.88  0.00010 ***
## rank4       -1.55146    0.41783   -3.71  0.00020 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 458.52  on 394  degrees of freedom
## AIC: 470.5
##
## Number of Fisher Scoring iterations: 4
```

- In the output above, the first thing we see is the call, this is R reminding us what the model we ran was, what options we specified, etc.

- Next we see the deviance residuals, which are a measure of model fit. This part of output shows the distribution of the deviance residuals for individual cases used in the model. Below we discuss how to use summaries of the deviance statistic to assess model fit.

- The next part of the output shows the coefficients, their standard errors, the z-statistic (sometimes called a Wald z-statistic), and the associated p-values. Both **gre** and **gpa** are statistically significant, as are the three terms for **rank**. The logistic regression coefficients give the change in the log odds of the outcome for a one unit increase in the predictor variable.

  - For every one unit change in **gre**, the log odds of admission (versus non-admission) increases by `0.002`.
  - For a one unit increase in **gpa**, the log odds of being admitted to graduate school increases by `0.804`.
  - The indicator variables for **rank** have a slightly different interpretation. For example, having attended an undergraduate institution with **rank** of 2, versus an institution with a **rank** of 1, changes the log odds of admission by `−0.675`.

- Below the table of coefficients are fit indices, including the null and deviance residuals and the AIC. Later we show an example of how you can use these values to help assess model fit.

We can use the **confint** function to obtain confidence intervals for the coefficient estimates. Note that for logistic models, confidence intervals are based on the profiled log-likelihood function. We can also get CIs based on just the standard errors by using the default method.

```
## CIs using profiled log-likelihood
confint(mylogit)
```

```
## Waiting for profiling to be done...
```

```
##                      2.5 %    97.5 %
## (Intercept) -6.271620 -1.79255
## gre          0.000138  0.00444
## gpa          0.160296  1.46414
## rank2       -1.300889 -0.05675
## rank3       -2.027671 -0.67037
## rank4       -2.400027 -0.75354
```

```
## CIs using standard errors
confint.default(mylogit)
```

```
##                     2.5 %   97.5 %
## (Intercept) -6.22424 -1.75572
## gre          0.00012  0.00441
## gpa          0.15368  1.45439
## rank2       -1.29575 -0.05513
## rank3       -2.01699 -0.66342
## rank4       -2.37040 -0.73253
```

We can test for an overall effect of **rank** using the **wald.test** function of the **aod** library. The order in which the coefficients are given in the table of coefficients is the same as the order of the terms in the model. This is important because the **wald.test** function refers to the coefficients by their order in the model. We use the **wald.test** function. **b** supplies the coefficients, while **Sigma** supplies the variance covariance matrix of the error terms, finally **Terms** tells R which terms in the model are to be tested, in this case, terms 4, 5, and 6, are the three terms for the levels of **rank**.

```
wald.test(b = coef(mylogit), Sigma = vcov(mylogit), Terms = 4:6)
```

```
## Wald test:
## ----------
##
## Chi-squared test:
## X2 = 20.9, df = 3, P(> X2) = 0.00011
```

The chi-squared test statistic of 20.9, with three degrees of freedom is associated with a p-value of 0.00011 indicating that the overall effect of **rank** is statistically significant.

We can also test additional hypotheses about the differences in the coefficients for the different levels of rank. Below we test that the coefficient for **rank**=2 is equal to the coefficient for **rank**=3. The first line of code below creates a vector **l** that defines the test we want to perform. In this case, we want to test the difference (subtraction) of the terms for **rank**=2 and **rank**=3 (i.e., the 4th and 5th terms in the model). To contrast these two terms, we multiply one of them by 1, and the other by -1. The other terms in the model are not involved in the test, so they are multiplied by 0. The second line of code below uses **L=l** to tell R that we wish to base the test on the vector **l** (rather than using the Terms option as we did above).

```
l <- cbind(0, 0, 0, 1, −1, 0)
wald.test(b = coef(mylogit), Sigma = vcov(mylogit), L = l)
```

```
## Wald test:
## ----------
##
## Chi-squared test:
## X2 = 5.5, df = 1, P(> X2) = 0.019
```

The chi-squared test statistic of 5.5 with 1 degree of freedom is associated with a p-value of 0.019, indicating that the difference between the coefficient for **rank**=2 and the coefficient for **rank**=3 is statistically significant.

You can also exponentiate the coefficients and interpret them as odds-ratios. R will do this computation for you. To get the exponentiated coefficients, you tell R that you want to exponentiate (**exp**), and that the object you want to exponentiate is called coefficients and it is part of mylogit (**coef(mylogit)**). We can use the same logic to get odds ratios and their confidence intervals, by exponentiating the confidence intervals from before. To put it all in one table, we use **cbind** to bind the coefficients and confidence intervals column-wise.

```
## odds ratios only
exp(coef(mylogit))
```

```
## (Intercept)          gre          gpa        rank2        rank3        rank4
##      0.0185       1.0023       2.2345       0.5089       0.2618       0.2119
```

```
## odds ratios and 95% CI
exp(cbind(OR = coef(mylogit), confint(mylogit)))
```

```
## Waiting for profiling to be done...
```

```
##                   OR    2.5 % 97.5 %
## (Intercept) 0.0185 0.00189  0.167
## gre         1.0023 1.00014  1.004
## gpa         2.2345 1.17386  4.324
## rank2       0.5089 0.27229  0.945
## rank3       0.2618 0.13164  0.512
## rank4       0.2119 0.09072  0.471
```

Now we can say that for a one unit increase in **gpa**, the odds of being admitted to graduate school (versus not being admitted) increase by a factor of 2.23. For more information on interpreting odds ratios see our FAQ page How do I interpret odds ratios in logistic regression? (https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faq-how-do-i-interpret-odds-ratios-in-logistic-regression/) . Note that while R produces it, the odds ratio for the intercept is not generally interpreted.

You can also use predicted probabilities to help you understand the model. Predicted probabilities can be computed for both categorical and continuous predictor variables. In order to create predicted probabilities we first need to create a new data frame with the values we want the independent variables to take on to create our predictions.

We will start by calculating the predicted probability of admission at each value of rank, holding **gre** and **gpa** at their means. First we create and view the data frame.

```
newdata1 <- with(mydata, data.frame(gre = mean(gre), gpa = mean(gpa), rank = fac
tor(1:4)))

## view data frame
newdata1
```

```
##    gre  gpa rank
## 1 588 3.39    1
## 2 588 3.39    2
## 3 588 3.39    3
## 4 588 3.39    4
```

*These objects must have the same names as the variables in your logistic regression above (e.g. in this example the mean for **gre** must be named gre).* 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. The first line of code below is quite compact, we will break it apart to discuss what various components do. The **newdata1$rankP** tells R that we want to create a new variable in the dataset (data frame) **newdata1** called **rankP**, the rest of the command tells R that the values of **rankP** should be predictions made using the **predict( )** function. The options within the parentheses tell R that the predictions should be based on the analysis **mylogit** with values of the predictor variables coming from **newdata1** and that the type of prediction is a predicted probability (**type="response"**). The second line of the code lists the values in the data frame **newdata1**. Although not particularly pretty, this is a table of predicted probabilities.

```
newdata1$rankP <- predict(mylogit, newdata = newdata1, type = "response")
newdata1
```

```
##    gre  gpa rank rankP
## 1 588 3.39    1 0.517
## 2 588 3.39    2 0.352
## 3 588 3.39    3 0.219
## 4 588 3.39    4 0.185
```

In the above output we see that the predicted probability of being accepted into a graduate program is 0.52 for students from the highest prestige undergraduate institutions (**rank**=1), and 0.18 for students from the lowest ranked institutions (**rank**=4), holding **gre** and **gpa** at their means. We can do something very similar to create a table of predicted probabilities varying the value of **gre** and **rank**. We are going to plot these, so we will create 100 values of **gre** between 200 and 800, at each value of rank (i.e., 1, 2, 3, and 4).

```
newdata2 <- with(mydata, data.frame(gre = rep(seq(from = 200, to = 800, length.o
ut = 100),
    4), gpa = mean(gpa), rank = factor(rep(1:4, each = 100))))
```

The code to generate the predicted probabilities (the first line below) is the same as before, except we are also going to ask for standard errors so we can plot a confidence interval. We get the estimates on the link scale and back transform both the predicted values and confidence limits into probabilities.

```
newdata3 <- cbind(newdata2, predict(mylogit, newdata = newdata2, type = "link",
    se = TRUE))
newdata3 <- within(newdata3, {
    PredictedProb <- plogis(fit)
    LL <- plogis(fit - (1.96 * se.fit))
    UL <- plogis(fit + (1.96 * se.fit))
})


## view first few rows of final dataset
head(newdata3)
```

```
##   gre  gpa rank    fit se.fit residual.scale    UL    LL PredictedProb
## 1 200 3.39    1 -0.811  0.515              1 0.549 0.139         0.308
## 2 206 3.39    1 -0.798  0.509              1 0.550 0.142         0.311
## 3 212 3.39    1 -0.784  0.503              1 0.551 0.145         0.313
## 4 218 3.39    1 -0.770  0.498              1 0.551 0.149         0.316
## 5 224 3.39    1 -0.757  0.492              1 0.552 0.152         0.319
## 6 230 3.39    1 -0.743  0.487              1 0.553 0.155         0.322
```
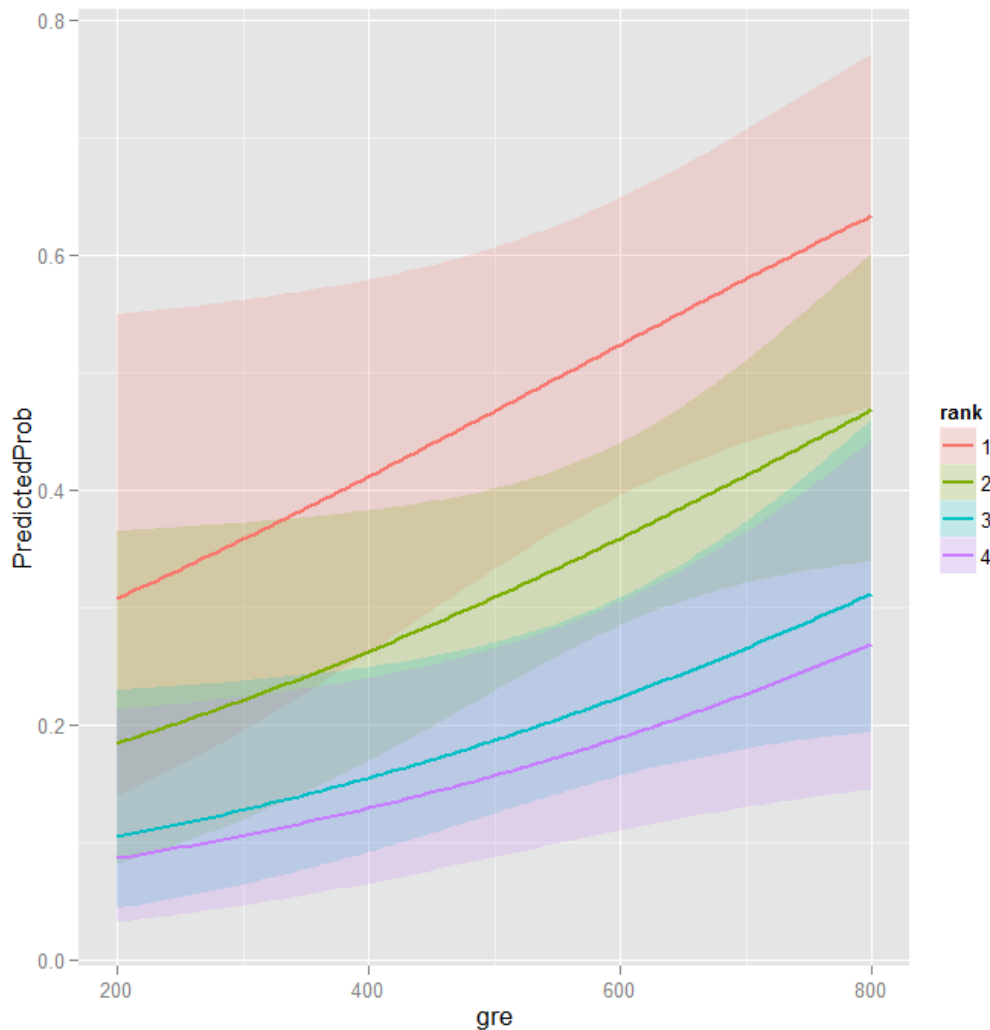
It can also be helpful to use graphs of predicted probabilities to understand and/or present the model. We will use the **ggplot2** package for graphing. Below we make a plot with the predicted probabilities, and 95% confidence intervals.

```
ggplot(newdata3, aes(x = gre, y = PredictedProb)) + geom_ribbon(aes(ymin = LL,
    ymax = UL, fill = rank), alpha = 0.2) + geom_line(aes(colour = rank),
    size = 1)
```

We may also wish to see measures of how well our model fits. This can be particularly useful when comparing competing models. The output produced by **summary(mylogit)** included indices of fit (shown below the coefficients), including the null and deviance residuals and the AIC. One measure of model fit is the significance of the overall model. This test asks whether the model with predictors fits significantly better than a model with just an intercept (i.e., a null model). The test statistic is the difference between the residual deviance for the model with predictors and the null model. The test statistic is distributed chi-squared with degrees of freedom equal to the differences in degrees of freedom between the current and the null model (i.e., the number of predictor variables in the model). To find the difference in deviance for the two models (i.e., the test statistic) we can use the command:

```
with(mylogit, null.deviance - deviance)
```

```
## [1] 41.5
```

The degrees of freedom for the difference between the two models is equal to the number of predictor variables in the mode, and can be obtained using:

```
with(mylogit, df.null - df.residual)
```

```
## [1] 5
```

Finally, the p-value can be obtained using:

```
with(mylogit, pchisq(null.deviance - deviance, df.null - df.residual, lower.tail
 = FALSE))
```

```
## [1] 7.58e-08
```

The chi-square of 41.46 with 5 degrees of freedom and an associated p-value of less than 0.001 tells us that our model as a whole fits significantly better than an empty model. This is sometimes called a likelihood ratio test (the deviance residual is -2*log likelihood). To see the model's log likelihood, we type:

```
logLik(mylogit)
```

```
## 'log Lik.' -229 (df=6)
```

Things to consider

- 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.
- Separation or quasi-separation (also called perfect prediction), a condition in which the outcome does not vary at some levels of the independent variables. See our page FAQ: What is complete or quasi-complete separation in logistic/probit regression and how do we deal with them? (https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faqwhat-is-complete-or-quasi-complete-separation-in-logisticprobit-regression-and-how-do-we-deal-with-them/) for information on models with perfect prediction.
- Sample size: Both logit and probit models require more cases than OLS regression

because they use maximum likelihood estimation techniques. It is sometimes possible to estimate models for binary outcomes in datasets with only a small number of cases using exact logistic regression. It is also important to keep in mind that when the outcome is rare, even if the overall dataset is large, it can be difficult to estimate a logit model.

- Pseudo-R-squared: Many different measures of psuedo-R-squared exist. They all attempt to provide information similar to that provided by R-squared in OLS regression; however, none of them can be interpreted exactly as R-squared in OLS regression is interpreted. For a discussion of various pseudo-R-squareds see Long and Freese (2006) or our FAQ page What are pseudo R-squareds? (https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faq-what-are-pseudo-r-squareds/)

- Diagnostics: The diagnostics for logistic regression are different from those for OLS regression. For a discussion of model diagnostics for logistic regression, see Hosmer and Lemeshow (2000, Chapter 5). Note that diagnostics done for logistic regression are similar to those done for probit regression.

References

Hosmer, D. & Lemeshow, S. (2000). Applied Logistic Regression (Second Edition). New York: John Wiley & Sons, Inc.

Long, J. Scott (1997). Regression Models for Categorical and Limited Dependent Variables. Thousand Oaks, CA: Sage Publications.

See also

- R Online Manual **glm** (http://stat.ethz.ch/R-manual/R-patched/library/stats/html/glm.html)
- Stat Books for Loan, Logistic Regression and Limited Dependent Variables (/books/#Logistic Regression and Related Methods)
- Everitt, B. S. and Hothorn, T. A Handbook of Statistical Analyses Using R (http://cran.r-

project.org/web/packages/HSAUR/)

Click here to report an error on this page or leave a comment

How to cite this page (https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faq-how-do-i-cite-web-pages-and-programs-from-the-ucla-statistical-consulting-group/)