Interpretation and Prediction of a Logistic Model

Joseph M Hilbe
hilbe@asu.edu: 11 March, 2014

Logistic regression is foremost used to model a binary (0/1) variable based on one or more other variables, called predictors. The binary variable being modeled is generally referred to as the response variable, and I refer to it as such here. For a model to fit the data well, it is assumed that the predictors are uncorrelated with one another, that they are significantly related to the response, and that the observations or data elements of a model are also uncorrelated. The response is also assumed to fit closely to an underlying probability distribution from which the response is a theoretical sample. The goal of a model is to estimate the true parameters of the underlying PDF of the model based on the response as adjusted by its predictors. In the case of logistic regression, the response is binary (0/1) and follows a Bernoulli probability distribution. Since the Bernoulli distribution is a subset of the more general binomial distribution, logistic regression is recognized as a member of the binomial family of regression models. A comprehensive analysis of these relationships is provided in Hilbe (2009).

In this short monograph I assume that the reader is familiar with the basics of regression. I address only the fundamentals of interpreting a logistic model, and on how to predict the probability that the response has a value of 1 given a specific set of predictor values. Interpretation of logistic model coefficients usually involves their exponentiation, which allows them to be understood as odds ratios. This capability is unique to the class of logistic models. The fact that a logistic model can be used to assess the odds ratio of predictors, and also can be used to determine the probability of the response occurring based on specific predictor values, called covariate patterns, is the prime reason it has enjoyed such popularity in the statistical community for the past several decades.

I shall use the **medpar** data to demonstrate how a simple binary response (0,1) logistic model can be interpreted. The **medpar** data can be found in my **COUNT** package on CRAN, as well as in the data packages associated with most of my books . I only present the basics here, but this explanation should give you a sense for how models estimated using logistic regression can be interpreted. The modeling is done using R. I have used R version 3.0.1. However, if you are using an older version of R you should still get the same results.

I am assuming that you have installed the packages that are used in this monograph. Look for all library functions and be sure that they have been installed prior to replicating the example shown here. Of course, it is not necessary to duplicate what I have done - you can use another example following the same logic.

I want to emphasize that I am not discussing the more detailed methods that can be used to check or manipulate a model to effect a more satisfactory model. This model and explanation is for pedagogical purposes only.

The **medpar** data is from the US national database on hospital records for Medicare patients. The data comes from the 1991 Arizona component of the MedPar database, and represents data from one DRG, or Diagnostic Related Group. Hospitals are reimbursed by the federal government based on the type of DRG. I first constructed the data set to use when conducting workshops for the Health Care Financing System (HCFA) in 1992. All variables or information in the data that could be used to violate the privacy rights of the patients whose hospital records

are part of the **medpar** data set were excluded. The variables included in the example I use in this monograph are described as

response : *died* 1=patient died within 48 hours of admission; 0=not die
predictors: *white* 1=patient identifies themselves as white; 0 = as non-white
hmo 1= patient belongs to a Health Maintenance Organization; 0=not belong
type 1=elective admission; 2= urgent admission; 3=emergency admission

others not included in example model
los length of stay in hospital
age80 1=patient age 80 and over; 0=younger than 80.

We begin by loading the **medpar** data from the COUNT package, which consists of 1495 observations.

```
> library(COUNT)
> data(medpar)
> head(medpar)    # displays the values for the first 6 observations
```

```
  los hmo white died age80 type type1 type2 type3 provnum
1   4   0     1    0     0    1     1     0     0  030001
2   9   1     1    0     0    1     1     0     0  030001
3   3   1     1    1     1    1     1     0     0  030001
4   9   0     1    0     0    1     1     0     0  030001
5   1   0     1    1     1    1     1     0     0  030001
6   4   0     1    1     0    1     1     0     0  030001
```

I model *died* on *white*, *hmo*, and levels of *type*, with level 1 as the reference level. I have combined two functions -- the **glm** function and **summary** function. **glm** estimates the parameters and provides associated statistics. The **summary** function prints the model results to the screen in a fairly standard manner. Confidence intervals are not displayed. Another function is needed to calculate and display them afterwards.

```
> summary(mymodel <- glm(died ~ white + hmo + factor(type),
+                                  data=medpar,
+                                  family=binomial)
+ )

Call:
glm(formula = died ~ white + hmo + factor(type), family = binomial,
    data = medpar)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.1594  -0.8853  -0.8853   1.4795   1.6615

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   -1.09058    0.20535  -5.311 1.09e-07 ***
white          0.35617    0.20712   1.720  0.08551 .
hmo            0.04758    0.15032   0.317  0.75161
factor(type)2  0.33963    0.14217   2.389  0.01690 *
factor(type)3  0.64421    0.21582   2.985  0.00284 **
```

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)
    Null deviance: 1922.9  on 1494  degrees of freedom
Residual deviance: 1907.9  on 1490  degrees of freedom
AIC: 1917.9

Number of Fisher Scoring iterations: 4
```

Interpretation of the coefficients of a logistic regression differ from the interpretation of coefficients from a normal linear regression model. A coefficient of a traditional linear regression predictor is interpreted as the change in the value of the response given a one unit increase in value of the predictor. The coefficients, or slopes, of a logistic regression model indicate, on the other hand, the change in value of the log-odds of the response given a one-unit increase in value of a predictor. For binary (0/1) predictors, the one-unit increase is from level 0 to level 1. For a categorical predictor, the one-unit change is from the specified reference level to the level of the predictor associated with the coefficient. For a continuous predictor the one-unit change is from the lower of two contiguous values to the next higher value.

Referring to the model output above, note that the *p*-values of *white* (0.08551) and *hmo* (0.75161) are both greater than 0.05. This tells us that the predictors do NOT significantly contribute to the understanding of the response, *died*. Both levels of *type* are significant; i.e., they contribute to patient deaths in the hospital. *type1* (an elective hospital admission) is the reference. *type2* is an urgent admit and *type3* is an emergency admission. If *white* were significant, we could interpret it as meaning that when comparing white patients and non-white patients, being white adds 0.356 points to the log odds of death. **White patients have a .365 greater log-odds of death in the hospital compared to non-white patients.** For *type*, an urgent admission (*type2*) results in a 0.34 increase in the log-odds of death compared to a patient who was admitted as an elective admission. Patients admitted as emergency patients increase the log-odds of death by 0.64 compared to an elective admission. Again, because the *p*-value of the *type* predictors are less than 0.05, they are significant (at the 0.05 level - sometimes called the "*alpha*=0.05 level", or α=0.05).

We now exponentiate the coefficients, which are *odds ratios*. This is the preferred way to interpret logistic parameter estimates. Reparameterizing the model in terms of odds ratios has no bearing at all on the significance of the predictors, just how they are interpreted.

```
> exp(coef(mymodel))
  (Intercept)         white           hmo factor(type)2 factor(type)3
    0.3360218     1.4278500     1.0487290     1.4044229     1.9044888
```

INTERPRETATION OF ODDS RATIOS

The levels of *type* are the only predictors that are significant. The model tells us that urgent admit patients have some 40% greater odds of death (1.4044) compared to elective patients, holding other predictor values constant. Emergency patients have some 90% greater odds of death (1.9045) compared to elective patients, holding other values constant.

The confidence intervals of coefficients can also be used to assess significance. If a coefficient is 0, it has no impact or influence on the response. The confidence intervals inform us that if we repeated the analysis a very large number of times, the true coefficient would be within

the range of the lower and upper levels of the interval 95 times out of 100. **If the interval includes zero (0), the predictor is NOT significant**. This is important to remember. When assessing the significance of odds ratios, a predictor is not significant if the confidence interval includes one (1). For example, if the confidence interval of the odds ratio of a predictor is from 0.80 to 1.45, the predictor is not significant. The p-value of the predictor will also be greater than 0.05.

We use the **confint** function to obtain profile confidence intervals. These are confidence intervals based on the likelihood ratio test. You can obtain the standard "model" confidence intervals by using the **confint.default** function instead. These will be the same as what is provided when using Stata, SAS and other software. Both the "profile likelihood" confidence intervals and "model" confidence intervals are given below. Refer to Hilbe & Robinson (2013) for an analysis of profile likelihood and R code for its implementation.

```
> confint(mymodel)
Waiting for profiling to be done...
                    2.5 %     97.5 %
(Intercept)   -1.50483653 -0.6975616
white         -0.04101036  0.7731985    <= includes 0
hmo           -0.25026597  0.3396431    <= includes 0
factor(type)2  0.05929754  0.6171057
factor(type)3  0.21869574  1.0670262
```

Standard or "Model" confidence intervals

```
> confint.default(mymodel)
                    2.5 %     97.5 %
(Intercept)   -1.49305866 -0.6880998
white         -0.04978675  0.7621264    <= includes 0
hmo           -0.24704044  0.3421984    <= includes 0
factor(type)2  0.06096888  0.6182840
factor(type)3  0.22121121  1.0672161
```

Recall that to convert model coefficient values to odds ratios, the coefficient must be exponentiated. The same is the case for the confidence intervals of an odds ratio. Below are the odds ratios and both types of confidence intervals for the **titanicgrp** data.

ODDS RATIO

```
> exp(coef(mymodel))
  (Intercept)         white           hmo factor(type)2 factor(type)3
    0.3360218     1.4278500     1.0487290     1.4044229     1.9044888
```

ODDS RATIO PROFILE CONFIDENCE INTERVALS

```
> exp(confint(mymodel))
Waiting for profiling to be done...
                   2.5 %    97.5 %
(Intercept)   0.2220536 0.4977977
white         0.9598192 2.1666852     <= includes 1
hmo           0.7785937 1.4044462     <= includes 1
factor(type)2 1.0610909 1.8535556
factor(type)3 1.2444526 2.9067227
```

ODDS RATIO MODEL CONFIDENCE INTERVALS

```
> exp(confint.default(mymodel))
                    2.5 %    97.5 %
(Intercept)    0.2246844 0.5025301
white          0.9514323 2.1428278
hmo            0.7811091 1.4080397
factor(type)2 1.0628658 1.8557408
factor(type)3 1.2475869 2.9072746
```

Compare the model output, including model confidence intervals for odds ratios with Stata output. They are numerically identical. Note that I use the *nohead* option with Stata's **glm** command so that only the table of estimates is displayed. The *eform* option is given to produce a table of odds ratios and associated statistics. *nolog* inhibits a display of the iteration log.

```
. glm died white hmo i.type, fam(bin) nolog nohead eform

------------------------------------------------------------------------------
             |                 OIM
        died | Odds Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       white |    1.42785   .2957429     1.72   0.086      .951432    2.142828
         hmo |   1.048729   .1576437     0.32   0.752     .7811091     1.40804
             |
        type |
Urgent Admit |   1.404423   .1996736     2.39   0.017     1.062866    1.855741
Emergency Ad |   1.904489   .4110297     2.98   0.003     1.247587    2.907275
             |
       _cons |   .3360218   .0690023    -5.31   0.000     .2246843    .5025302
------------------------------------------------------------------------------
```

To re-iterate, odds ratios and their confidence intervals are obtained by exponentiation of the model coefficients and their standard errors. Profile confidence intervals of odds ratios are obtained the same way. However, the standard errors of odds ratios are not obtained by exponentiation of the model standard errors. Rather, statisticians use what is called the **delta method** for calculating the standard errors of odds ratios. In the case of logistic regression it is simple, and involves multiplying the model standard errors by the associated odds ratios. In R you must do this by hand; there is no built-in method using **glm** to display a table of odds ratios, its standard error, and other related statistics. The standard errors of the *mymodel* odds ratios may be calculated using the delta method as,

```
> OR <- exp(coef(mymodel))                  # vector of model odds ratios
> stderr <- sqrt(diag(vcov(mymodel)))       # model SEs
> #  stderr <- coef(summary(mymodel))[,2]     alternative calculation of SEs
> ORsea <- OR*stderr
> ORsea
  (Intercept)         white           hmo factor(type)2 factor(type)3
   0.06900222    0.29574271    0.15764369    0.19967359    0.41102971
```

The above values are identical with the standard errors of the Stata odds ratios.

The data may have more correlation in it than is allowed under the assumptions of the binomial probability function. Probability functions assume that each observation is independent of all other observations. Given that patients come from difference providers (hospitals), it may be the case that the relationship of death to type of admission, as well as race and insurance status may be more similar within providers than between providers. When this is the case, the

data may be correlated, and the standard error biased. When this happens a predictor may appear to significantly contribute to model when it in fact does not. We need to check for this situation.

A standard way to both test, and adjust for, extra correlation in the data is to scale the standard errors by the Pearson dispersion statistic. The dispersion is the Pearson statistic divided by the residual degrees of freedom. The residual degrees of freedom is the number of observations in the model less predictors, including the intercept. Scaling involves multiplying the model standard errors by the square root of the dispersion.

The Pearson dispersion is not given in **glm** output; you need to calculate it. The Pearson statistic is such an important statistic in dealing with this class of models that this is a real fault with the software. In any case, we may calculate the Pearson statistic (*PR*), the dispersion statistic (*disp*), model standard errors (*stderr*), and scaled standard errors (*scalese*) using the following code.

```
> PR <- resid(mymodel, type = "pearson")
> N <- nrow(medpar)
> P <- length(coef(mymodel))
> disp <- sum(PR^2) / (N - P)
> stderr <- coef(summary(mymodel))[,2]
> stderr
  (Intercept)         white           hmo factor(type)2 factor(type)3
    0.2053504     0.2071245     0.1503188     0.1421748     0.2158215
> scalese <- stderr*sqrt(disp)
> scalese
  (Intercept)         white           hmo factor(type)2 factor(type)3
    0.2056617     0.2074384     0.1505466     0.1423903     0.2161486
```

The R model for a logistic regression with scaled standard errors is shown below. Compare the scaled standard errors we calculated by hand with the standard errors of the *quasibinomial* model. They are the same.

```
> summary(scaledlogit <- glm(died ~ white + hmo + factor(type),
+                                data=medpar,
+                                family=quasibinomial))

Call:
glm(formula = died ~ white + hmo + factor(type), family = quasibinomial,
    data = medpar)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.1594  -0.8853  -0.8853   1.4795   1.6615

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   -1.09058    0.20566  -5.303 1.31e-07 ***
white          0.35617    0.20744   1.717  0.08619 .
hmo            0.04758    0.15055   0.316  0.75202
factor(type)2  0.33963    0.14239   2.385  0.01720 *
factor(type)3  0.64421    0.21615   2.980  0.00293 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for quasibinomial family taken to be 1.003034)
```

```
    Null deviance: 1922.9  on 1494  degrees of freedom
Residual deviance: 1907.9  on 1490  degrees of freedom
AIC: NA
```

Finally, another standard way of dealing with correlated data is by applying a *robust* or *sandwich* variance estimator to the model standard errors. That is, the standard errors are adjusted by a sandwich or robust estimator. Many statisticians, including myself, recommend that applying robust estimators for the standard errors should be the default manner of displaying model output. If the data is not correlated, or otherwise biased, then the adjustment is extinguished and no difference exists between them.

In order to develop a model with robust standard errors the *sandwich* library must be installed and loaded. The model is then run as usual, followed by a line that converts model standard errors to sandwich errors.

```
> library(sandwich)
> summary(roblogit <- glm(died ~ white + hmo + factor(type),
                          family=binomial,
                          data=medpar))
> sqrt(diag(vcovHC(roblogit, type = "HC0")))
  (Intercept)         white           hmo factor(type)2 factor(type)3
    0.2028963     0.2054485     0.1508660     0.1421628     0.2158494
```

Note that software packages at times calculate sandwich standard errors using slightly different algorithms. For example, the robust or sandwich standard errors given for the above R model are close to those displayed in Stata output, with a digit difference occurring at the ten-thousand's place - certainly nothing to be concerned with. SAS output differs slightly from both R and Stata.

```
. glm died white hmo i.type, fam(bin) nolog vce(robust)

------------------------------------------------------------------------------
             |               Robust
        died |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       white |   .3561698   .2055172     1.73   0.083    -.0466366    .7589762
         hmo |    .047579   .1509165     0.32   0.753    -.2482119    .3433699
             |
        type |
Urgent Admit |   .3396264   .1422104     2.39   0.017     .0608992    .6183537
Emergency Admit |   .6442136   .2159216     2.98   0.003     .2210151    1.067412
             |
       _cons |  -1.090579   .2029642    -5.37   0.000    -1.488382   -.6927766
------------------------------------------------------------------------------
```

Again, *white* and *hmo* are seen to not contribute to the model. Because of this fact, if we exclude *hmo*, for instance, from the model, the other standard errors and their associated *p*-values remain very close to, or are very nearly the same, in value. And again, scaling or applying a robust estimator to the standard errors does not change the values of their associated coefficients, or odds ratios.

PREDICTION

The linear predictor is obtained using the post-estimation `predict` function. It is defined as

$$\eta_i = x'\beta = \sum_{i=1}^{n} \beta_0 + x'_{1i}\beta_1 + x'_{2i}\beta_2 + \cdots + x'_{jn}\beta_j$$

where x'β is the matrix form of the statistic. The relationship of the linear predictor, η, or x'β, and the predicted probability, which is also called the fitted value, is:

$$\eta_i = log\left(\frac{\mu_i}{1-\mu_i}\right) = \sum_{i=1}^{n_i} \beta_0 + x'_{i,1}\beta_1 + x'_{i,2}\beta_2 + \cdots + x'_{i,n}\beta_n$$

log(μ/(1-μ)) is referred to as the *logistic link function*, or *logit* function. μ is the fitted value, and is defined as the log of the odds ; i.e., μ /(1− μ). The inverse of this relationship is

$$\mu = \frac{1}{1+\exp(-\eta)} \; or \; \frac{exp(\eta)}{1+exp(\eta)}$$

which is termed the *logistic inverse link* function. The linear predictor may be calculated for the above model as

```
> eta <- predict(mymodel)
> summary(eta)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1.09100 -0.73440 -0.73440 -0.65550 -0.68680 -0.04262
```

The fitted values, or probability that `died==1`, rests between 0 and 1. We may obtain them as

```
> mu <- 1/(1+exp(-eta))
> summary(mu)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.2515  0.3242  0.3242  0.3431  0.3347  0.4893
```

We may also obtain the fitted or predicted probability that `died==1` by using the saved statistic `fitted.value`. We may obtain the values using the code,

```
fit <- mymodel$fitted.value
summary(fit)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.2515  0.3242  0.3242  0.3431  0.3347  0.4893
```

Note that the predicted probabilities are identical in value to the by-hand method used earlier.

The final major point to discuss is the confidence interval surrounding the fitted value, or probability. It cannot be obtained as simple as it is when dealing with a normal linear regression. For a logistic model we must accommodate the nonlinearity that is implicit in the model.

The key point to remember when calculating the standard error of the prediction is that it should initially be based on the linear predictor, or eta, and then converted to the probability scale.

Calculating the standard error of the linear predictor first will help establish their Gaussian form, which better carries through when the inverse logistic link function maps them from the linear predictor scale to the probability or response scale.

I will demonstrate how to calculate the confidence intervals of μ (mu), the predicted probability that *died*==1, using the *mymodel* object we estimated earlier. Recall that it was obtained using the following code.

```
mymodel <- glm(died ~ white + hmo + factor(type), family=binomial, data=medpar)
```

We next calculate the standard error of the linear predictor. We use the *predict* function with the `type="link"` and `se.fit=TRUE` options to place the predictions on the scale of the linear predictor, and to guarantee that the *lpred* object is in fact the standard error of the linear prediction.

```
lpred <- predict(mymodel, newdata=medpar, type="link", se.fit=TRUE)
```

Now we calculate the confidence interval of the linear predictor. We use a 95% confidence interval, which is standard in for health care and social science research. We assume that both sides of the distribution are used in determining the confidence interval, which means that 0.025 is taken from each tail of the distribution. In terms of the normal distribution, we see that

```
 > qnorm(0.975) ### 0.975 is the 2.5% upper tail, with the same from the lower tail
[1] 1.959964

up <- lpred$fit + (1.96 * lpred$se.fit)
lo <- lpred$fit - (1.96 * lpred$se.fit)
eta <- lpred$fit
```

We may use the inverse logistic link function, to convert the above three statistics to the probability scale. We could also use the true inverse logit link function, exp(x)/(1+exp(x)) or 1/(1+exp(-x)) , to convert these to the probability scale. It is easier to simply use the linkinv fcuntion. A summary of each is displayed based on the following code.

```
upci <- mymodel$family$linkinv(up)
mu <- mymodel$family$linkinv(eta)
loci <- mymodel$family$linkinv(lo)
summary(loci)
summary(mu)
summary(upci)
```

```
> summary(loci)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1796  0.2946  0.2946  0.2967  0.2946  0.3790
> summary(mu)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.2515  0.3242  0.3242  0.3431  0.3347  0.4893
> summary(upci)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.3345  0.3553  0.3553  0.3931  0.3975  0.6108
```

To contrast R, Stata has an option to its predict command that specifically calculates the standard error of the prediction. We may obtain the same statistics using the Stata code below

```
use medpar
glm died hmo white i.type, family(bin) nolog
```

```
predict eta, xb                  // linear predictor; eta
predict se_eta, stdp             // standard error of the prediction
gen mu = exp(eta)/(1+exp(eta))   // or: predict mu
gen low = eta - invnormal(0.975) * se_eta
gen up  = eta + invnormal(0.975) * se_eta
gen lci = exp(low)/(1+exp(low))
gen uci = exp(up)/(1+exp(up))

sum lci mu uci
```

```
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         lci |      1495    .2966823    .0392547   .1796257   .3789932
          mu |      1495    .3431438    .0478897   .2515092   .4893474
         uci |      1495    .3930506    .0635593    .334456   .6107746
```

A plot of the predicted probability on a continuous predictor such as "length of hospital stay", or *los*, may be displayed together with the surrounding confidence interval. Following the model,

```
grlos <- glm(died ~ los, family=binomial, data=medpar)
```

we may use the same code as above to calculate the predicted probability and confidence intervals. We need only change the name of *mymodel* to *grlos*; e.g.,

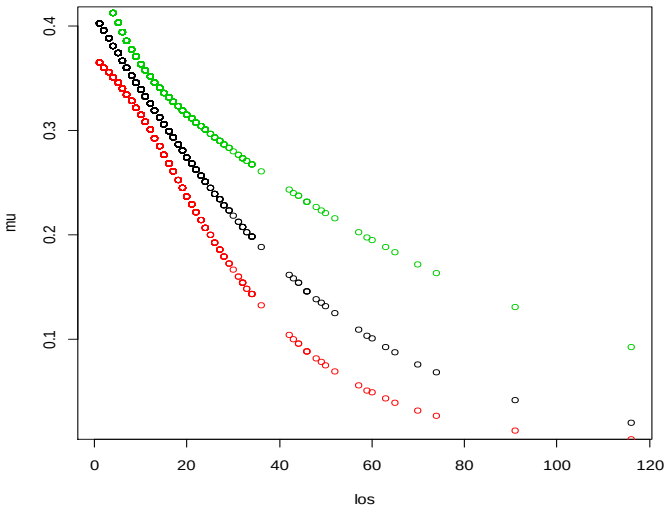
```
lpred <- predict(grlos, newdata=medpar, type="link", se.fit=TRUE)
```

The fit (mu) and confidence intervals are calculated as,

```
> summary(loci)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.004015 0.293000 0.328700 0.312900 0.350900 0.364900
> summary(mu)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01988 0.31910 0.35310 0.34310 0.38140 0.40320
> summary(upci)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.09265 0.34640 0.37820 0.37540 0.41280 0.44260
```

A simple R plot of the predicted probability of death for day in the hospital for patients in this data is displayed as:

```
layout(1)
plot(los, mu, col=1)
lines(los, loci, col=2, type='p')
lines(los,upci, col=3, type='p')
```



Using the Stata code

```
. scatter mu lci uci los
```

produces the same simple figure.

A summary of the functions used in the above discussion can be found in the table below. You can paste them into your *New Script* editor, which is the second selection in the main *Files* menu.

Additional strategies and code for developing logistic regression functions independent of the **glm** may be found in Hilbe & Robinson (2013). Another very good way to learn about developing R code for generalized linear models (GLM) and generalized linear mixed models (GLMM) from both a traditional frequency-based and from a Bayesian perspective using JAGS, see Zuur, Hilbe, and Ieno (2013). Note that modeling data using a logistic regression entails much more evaluation than presented here. I have only touched on the basics of model interpretation and prediction. I recommend Hilbe (2009) for a comprehensive analysis and guideline for all aspects of modeling this class of data. The book also discusses a number of extensions to the basic logistic model.

R *FUNCTIONS FOR BINARY LOGISTIC REGRESSION*

```
============================================================
library(COUNT)
data(medpar)
attach(medpar)
summary(mymodel <- glm(died ~ white + hmo + factor(type),
                       data=medpar,
                       family=binomial))
exp(coef(mymodel))              # display coefficients to odds ratio
confint(mymodel)                # profile-based confidence intervals
confint.default(mymodel)        # model-based confidence intervals

# odds ratio CI
exp(confint(mymodel))           # confidence intervals of odds ratios

# odds ratio SE
OR <- exp(coef(mymodel))                  # vector of model odds ratios
```

```
stderr <- sqrt(diag(vcov(mymodel)))       # model SEs
#  stderr <- coef(summary(mymodel))[,2]     alternative calculation of SE
ORse <- OR*stderr                         # SE of odds ratios
ORse

# scaled SEs
PR <- resid(mymodel, type = "pearson")    # Pearson residuals
N <- nrow(medpar)                         # observations in model
P <- length(coef(mymodel))                # number of model predictors
disp <- sum(PR^2) / (N - P)               # Pearson dispersion
stderr <- coef(summary(mymodel))[,2]      # model SEs
scalese <- stderr*sqrt(disp)              # scaled SEs
scalese

summary(scaledlogit <- glm(died ~ white + hmo + factor(type),
                                  data=medpar,
                                  family=quasibinomial))

# robust or sandwich SEs
library(sandwich)
summary(roblogit <- glm(died ~ white + hmo + factor(type),
                               family=binomial,
                               data=medpar))
sqrt(diag(vcovHC(roblogit, type = "HC0")))

# linear predictor, or logit
summary(eta <- predict(mymodel))

# fitted value, or predicted probability that died==1
fit <- mymodel$fitted.value
summary(fit)

# model of died on predictors, calc fit and its CI
lpred <- predict(mymodel, newdata=medpar, type="link", se.fit=TRUE)
up <- lpred$fit + (1.96 * lpred$se.fit)
lo <- lpred$fit - (1.96 * lpred$se.fit)
eta <- lpred$fit
upci <- mymodel$family$linkinv(up)
mu <- mymodel$family$linkinv(eta)
loci <- mymodel$family$linkinv(lo)
summary(loci)
summary(mu)
summary(upci)

# model of died on los, predicting fit and CI of fit
grlos <- glm(died ~ los, family=binomial, data=medpar))
lpred <- predict(grlos, newdata=medpar, type="link", se.fit=TRUE)
up <- lpred$fit + (1.96 * lpred$se.fit)
lo <- lpred$fit - (1.96 * lpred$se.fit)
eta <- lpred$fit
upci <- grlos$family$linkinv(up)
mu <- grlos$family$linkinv(eta)
loci <- grlos$family$linkinv(lo)
summary(loci)
summary(mu)
summary(upci)

# simple plot of predicted probability of died, and its CI
layout(1)
plot(los, mu, col=1)
lines(los, loci, col=2, type='p')
lines(los,upci, col=3, type='p')
================================================================
```

REFERENCES

Hilbe, Joseph M (2009), *Logistic Regression Models*, Boca Raton, FL: Chapman & Hall/CRC

Hilbe, J.M. and A.P. Robinson (2013), *Methods of Statistical Model Estimation*, Boca Raton, FL:Chapman & Hall/CRC

Zuur A.F., J.M Hilbe, and E.N. Ieno (2013), *A Beginner's Guide to GLM and GLMM with R: A frequentist and Bayesian perspective for ecologists*, Newburgh, UK: Highlands Statistics. http://www.highstat.com/BGGLM.htm