Logistic Regression

Hannah Waddel

5/3/2021

Logistic Regression in R

Fitting a logistic regression model in R is similar to fitting a linear regression model, but we are now going to use a different function: *glm()*.

GLM stands for generalized linear model. These models expand upon the SLR and MLR models we have used in the past to now include different types of outcome variables. A logistic regression model is a generalized linear model with a binary outcome.

We begin by loading the packages that we will need. Install any of the packages that you do not have on your computer already, and load with the *library()* function.

We will need the epiDisplay package to more easily do the likelihood ratio test and the car package for the type III partial tests. We will also need the multcomp ("multiple comparison") package in order to calculate the odds ratios between the different levels of sectors. DescTools contains the function to calculate our C-statistic (area under the ROC curve).

```r
library(epiDisplay)
library(car)
library(multcomp)
library(DescTools)
```

We begin by setting our working directory and reading in the data. To do this, we use the *read.table()* function. This is a very similar function to the *read.csv()* function that we have used in the past. Observe that we set the argument "header=TRUE". This tells R to use the first row of the dengue.dat file as column names.

```r
setwd("~/Work/BIOS 591P 2021/R Materials/4 Logistic Regression")
dengue <- read.table("dengue.dat",header=TRUE)
```

Let's check and make sure that we read it in correctly. The *head()* function prints the first few lines of the data table.

```r
head(dengue)
```

```
##    ID AGE SEX MOSNET DENGUE SCREEN SES SECTOR SECTOR1 SECTOR2 SECTOR3 SECTOR4
## 1 501  16   2      1      2      2   2      1       1       0       0       0
## 2 502  33   1      1      2      1   2      1       1       0       0       0
## 3 503   1   1      1      2      3   2      1       1       0       0       0
## 4 504  35   1      1      2      1   2      1       1       0       0       0
## 5 505   6   1      1      2      1   2      1       1       0       0       0
## 6 511  27   2      1      2      2   2      2       0       1       0       0
```

Looks good.

Dummy variables in R: Factors

To deal with categorical/dummy variables in R, we use a data type called factors. These do basically the same thing as the "CLASS(PARAM='ref',REF='X')". One category is chosen as a reference category. When we fit a model including a factor variable, R will automatically treat it as a reference-cell variable.

To turn a variable into a factor, we call the *as.factor()* function to convert it.

```r
as.factor(dengue$SECTOR)
```

```
##   [1] 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3
##  [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3
##  [75] 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [112] 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [149] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [186] 1 1 1 1 1 1 1 1 5 5 5
## Levels: 1 2 3 4 5
```

When we run *as.factor()*, all the data stays the same, but now we can see the "levels" of the factor. These are all the unique values of the variable.

We haven't saved the *as.factor()* output yet, so we will make a new column with the sector factor data. In R and SAS, it's best practice not to overwrite any data in your original dataset, but add new columns/variables as you manipulate the data. We will create a variable called "SECTOR_F" (sector factor). We can also verify that the new variable is a factor with the *is.factor()* function.

```r
dengue$SECTOR_F <- as.factor(dengue$SECTOR)

is.factor(dengue$SECTOR_F)
```

```
## [1] TRUE
```

In order to fit our logistic regression model, we will also need the outcome, dengue, to be a factor.

```r
dengue$DENGUE_F <- as.factor(dengue$DENGUE)
```

Fitting the logistic regression model

To fit the logistic regression model, we use the *glm()* function. Just like the *lm()* function, the *glm()* function takes a formula and a dataset as its first arguments. However, one more parameter is necessary, in order to tell R that we are fitting a logistic regression model specifically. This is the "family=binomial". This "family" parameter deals with some theoretical details of generalized linear models and logistic regression models, but it is basically telling R that we have a binary outcome variable.

```r
dengue.logistic <- glm(DENGUE_F ~ MOSNET + SECTOR_F + AGE,data=dengue,family=binomial)

summary(dengue.logistic)
```

```
##
## Call:
## glm(formula = DENGUE_F ~ MOSNET + SECTOR_F + AGE, family = binomial,
##     data = dengue)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4731  -1.0687   0.5087   0.8814   1.5671
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.120086   1.597958   2.578  0.00993 **
## MOSNET      -0.333525   1.271833  -0.262  0.79314
## SECTOR_F2   -1.561163   1.117435  -1.397  0.16238
## SECTOR_F3   -3.032164   1.078411  -2.812  0.00493 **
## SECTOR_F4   -2.750993   1.077382  -2.553  0.01067 *
## SECTOR_F5   -2.220018   1.072325  -2.070  0.03843 *
## AGE         -0.024263   0.009056  -2.679  0.00738 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 236.33  on 195  degrees of freedom
## Residual deviance: 203.71  on 189  degrees of freedom
## AIC: 217.71
##
## Number of Fisher Scoring iterations: 6
```

There is a problem with the fitted model. Observe that the coefficients for SECTOR_F are fit for "SECTOR_F2", "SECTOR_F3", "SECTOR_F4", and "SECTOR_F5". This is how R reports the coefficients for factors/categorical variables, and it is saying that these coefficients were fit for the variables "SECTOR_F equals 2", "SECTOR_F equals 3", etc. So Sector 1 is the reference category.

When we look at the levels of the SECTOR_F variable again, we see that the first level is 1. By default, R will take the first level as the reference category for dummy variable/reference cell coding.

```
levels(dengue$SECTOR_F)
```

```
## [1] "1" "2" "3" "4" "5"
```

```
levels(dengue$DENGUE_F)
```

```
## [1] "1" "2"
```

In addition, Dengue = 1 is the first level of DENGUE_F, so Dengue=1 is the reference category. R is modeling the probability, then, that Dengue=2, which is the reverse of what we want. We will also have to change the reference level for the dengue outcome variable.

We change the reference level for a factor with the *relevel()* function. The function arguments are data in the form of a factor and the desired new reference level.

We create the re-leveled factors for SECTOR and DENGUE and save them as new columns in our dataset.

```
dengue$SECTOR_F_RE <- relevel(dengue$SECTOR_F,ref=5)
dengue$DENGUE_F_RE <- relevel(dengue$DENGUE_F,ref=2)
```

We can see that our releveling worked by checking the levels of SECTOR_F_RE and DENGUE_F_RE, to make sure that the desired reference category is the first listed level.

```
levels(dengue$SECTOR_F_RE)
```

```
## [1] "5" "1" "2" "3" "4"
```

```
levels(dengue$DENGUE_F_RE)
```

```
## [1] "2" "1"
```

Looks good. Now we can fit the logistic regression model and get the desired results.

```
dengue.logistic <- glm(DENGUE_F_RE ~ MOSNET + SECTOR_F_RE + AGE,data=dengue,family=binomial)

summary(dengue.logistic)
```

```
##
## Call:
## glm(formula = DENGUE_F_RE ~ MOSNET + SECTOR_F_RE + AGE, family = binomial,
##     data = dengue)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5671  -0.8814  -0.5087   1.0687   2.4731
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.900068   1.325414  -1.434  0.15170
## MOSNET        0.333525   1.271833   0.262  0.79314
## SECTOR_F_RE1 -2.220018   1.072325  -2.070  0.03843 *
## SECTOR_F_RE2 -0.658855   0.553595  -1.190  0.23399
## SECTOR_F_RE3  0.812146   0.474987   1.710  0.08730 .
## SECTOR_F_RE4  0.530975   0.450196   1.179  0.23823
## AGE           0.024263   0.009056   2.679  0.00738 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 236.33  on 195  degrees of freedom
## Residual deviance: 203.71  on 189  degrees of freedom
## AIC: 217.71
##
## Number of Fisher Scoring iterations: 6
```

Overall Likelihood Ratio Test

To conduct the overall likelihood ratio test, we can use the *lrtest()* function from the epiDisplay package. The epiDisplay package has a lot of useful functions for epidemiology.

The likelihood ratio test in R needs one more argument. Recall that for the overall likelihood ratio test, we are comparing the full model:

$$\text{logit}(Pr(Y=1)) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 Z_1 + \beta_4 Z_2 + \beta_5 Z_3 + \beta_6 Z_4$$

To the reduced intercept-only model

$$\text{logit}(Pr(Y=1)) = \beta_0^*$$

So we will need to fit a reduced model for the logistic regression, and give the *lrtest()* function the full and reduced model as arguments.

To fit the intercept-only model in R, the formula will look a little different. The left side of the ~ will still have your outcome variable, but the right side/explanatory variables will just be 1. So our formula for an intercept-only model will be "DENGUE_F_RE ~ 1".

```
dengue.reduced <- glm(DENGUE_F_RE ~ 1, data=dengue,family=binomial)

lrtest(dengue.reduced,dengue.logistic)
```

```
## Likelihood ratio test for MLE method
## Chi-squared 6 d.f. =  32.62319 , P value =  1.238953e-05
```

Type III Test

To conduct the type III tests, we will need the *Anova()* function from car. Make sure that the function has a capital letter! We give the *Anova()* function the fitted logistic regression model, the type of partial test, and the desired method (in this case, Wald).

```
Anova(dengue.logistic,type=3,test="Wald")
```

```
## Analysis of Deviance Table (Type III tests)
##
## Response: DENGUE_F_RE
##             Df   Chisq Pr(>Chisq)
## (Intercept)  1  2.0551   0.151696
## MOSNET       1  0.0688   0.793137
## SECTOR_F_RE  4 13.6497   0.008501 **
## AGE          1  7.1778   0.007381 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

C Statistic

To calculate the C statistic (area under the ROC curve), we use the *Cstat()* function from the DescTools package.

```
Cstat(dengue.logistic)
```

```
## [1] 0.7368421
```

Odds ratios

R requires some statistical calculations by hand. In this case, R expects us to calculate the odds ratios as the exponential of the logistic regression coefficients β. See §7.2.9 in the class notes for more details. We get the coefficients from the model with the *coefficients()* function, then take the exponential with *exp()*.

```
exp(coefficients(dengue.logistic))
```

```
##  (Intercept)       MOSNET SECTOR_F_RE1 SECTOR_F_RE2 SECTOR_F_RE3 SECTOR_F_RE4
##    0.1495585    1.3958801    0.1086072    0.5174435    2.2527361    1.7005892
##          AGE
##    1.0245601
```

To get confidence intervals for the odds ratios, we use the *confint.default()* function to get confidence intervals for the coefficients, and then take the exponential of those confidence intervals. For a logistic regression model, we must use the *confint.default()* function to calculate confidence intervals in the style we have learned in this class, using the standard error. The *confint()* function uses a different method than *confint.default()* to calculate confidence intervals–it will still be right, but it will be slightly different.

```
exp(confint.default(dengue.logistic))
```

```
##                   2.5 %     97.5 %
## (Intercept)  0.01113312  2.0091185
## MOSNET       0.11541497 16.8823973
## SECTOR_F_RE1 0.01327681  0.8884298
## SECTOR_F_RE2 0.17484059  1.5313824
## SECTOR_F_RE3 0.88797477  5.7150495
## SECTOR_F_RE4 0.70370708  4.1096698
## AGE          1.00653434  1.0429086
```

Odds ratios for sectors

To get the odds ratios between each sector, and not just the odds ratios for 1 vs. 5, 2 vs. 5, etc., we will use the *glht()* function from the multcomp package. This stands for "general linear hypotheses test". This allows us to test and calculate various different hypotheses or estimates for our model coefficients. For example, if we are calculating the odds ratio for sector 2 and sector 4 in this model, we are testing $H_0 : \beta_4 = \beta_6$. But we don't need to worry about specifying those exact hypotheses.

The *glht()* function takes the fitted model as its first argument. The second argument is where we tell the *glht()* function that we are testing all pairs of the SECTOR_F_RE variable. We do this by specifying ‘mcp(SECTOR_F_RE=“Tukey”)’. The *mcp()* part is saying that we are doing a multiple comparison (in this case, comparing all the levels of sector). The argument ‘SECTOR_F_RE=“Tukey” ’ tells R that we are testing all pairs of the SECTOR_F_RE variable.

```
dengue.sector.test <- glht(dengue.logistic,mcp(SECTOR_F_RE="Tukey"))
```

We extract the estimated coefficient differences from the general linear hypothesis test with the *coefficient()* function, then exponentiate those estimates to get the estimated odds ratios for the pairs of sector.

```
exp(coefficients(dengue.sector.test))
```

```
##      1 - 5      2 - 5      3 - 5      4 - 5      2 - 1      3 - 1      4 - 1
##  0.1086072  0.5174435  2.2527361  1.7005892  4.7643596 20.7420601 15.6581700
##      3 - 2      4 - 2      4 - 3
##  4.3535883  3.2865214  0.7548995
```

We use the *confint.default()* function to get confidence interval for the coefficient differences, then take the exponential of those differences to get the odds ratio.

```
sector.or.ci <- exp(confint.default(dengue.sector.test))
```