Lab 10

Public Health 241: Statistical Analysis of Categorical Data

YOUR NAME / YOUR STUDENT ID HERE

TODAY'S DATE

The objective of this lab is to provide information regarding running logistic regressions in R. **This information should be useful for assignments 11 and 12, and for your final projects.**

1. Running Logistic Regressions: `glm`

The `glm` function helps you find the maximum likelihood estimates for your specified logistic model. The general form of this command is as follows (this is an **example**, don't try to run this command yet):

```
my.model <- glm(formula = ind.var ~ dep.var1 + dep.var2,
                family = binomial(link='logit'), data=my.data)
```

The call to `glm()` saves the model into the variable `my.model`. Here, the outcome (independent) variable is indicated by `ind.var`, and the dependent variable names following the `~` explain to the model which columns you are providing as your significant variables to the model. Finally, we must specify a type of outcome variable, which you can see in the example is a `binomial(link='logit')` variable. The order of the dependent variables does not matter. For example, take a look at the command below. The titanic dataset for this lab is loaded as `titanic`:

```
logit.titanic <- glm(survived ~ pclass + age,
                     family=binomial(link='logit'), data=titanic)
```

The command above corresponds to the following model:

$$\log\left(\frac{P(\text{survived}|\text{pclass, age})}{1-P(\text{survived}|\text{pclass, age})}\right) = a + b * \text{pclass} + c * \text{age}$$

or

$$\log\left(\frac{P(\text{survived}|\text{pclass, age})}{1-P(\text{survived}|\text{pclass, age})}\right) = \beta_0 + \beta_1 * \text{pclass} + \beta_2 * \text{age}$$

Let's take a look at the output of the glm function we called above by using another function call `summary()`.

```
summary(logit.titanic)
```

```
##
## Call:
## glm(formula = survived ~ pclass + age, family = binomial(link = "logit"),
##     data = titanic)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0363  -0.9057  -0.6635   1.0785   2.2794
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.114317   0.327127    9.52  < 2e-16 ***
```

```
## pclass      -1.101464   0.095532  -11.53  < 2e-16 ***
## age         -0.036992   0.005464   -6.77 1.29e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1414.6  on 1045  degrees of freedom
## Residual deviance: 1256.2  on 1043  degrees of freedom
##   (263 observations deleted due to missingness)
## AIC: 1262.2
##
## Number of Fisher Scoring iterations: 4
```

Here's how we should interpret the output above:

- **Deviance** is a measure of badness of fit - higher numbers indicate worse fit. Every data point in the model has a deviance associated with it, which is calculated, and a non-parametric description is shown below
- **Coeffcients**: This row contains point estimates for your model coefficients, including an estimate of your intercept.
- `Estimate`: For example, the estimate of our intercept is 3.114, the estimate of the coefficient on `pclass` is -1.101, and the estimate of the coefficient for `age` is -0.3699.
- `Std. Error`: This is self-explanatory. This is an estimate of how much, on average, the current estimates would move if the study were re-run repeatedly with different data.
- `z value`: This is the quotient of the estimate by the standard error
- `Pr(>|z|)`: lists the tailed p-values that correspond to those z-values in a standard normal distribution.
- There are two forms of deviance:
  - *Null deviance* shows how well the outcome variable is predicted by a model that includes only the intercept. i.e. The null hypothesis of this model is that all of your covariate coefficients are zero.
  - *Residual deviance* is different in that it applies all weights and displacements. This is the one that shows in the description.
- The **Akaike Information Criterion (AIC)** allows you a method for assessing the quality of your model in comparison to other models that you have created that are more or less complex. You should choose the smallest AIC when comparing.
- **Fisher scoring** maximizes the likelihood by iteratively getting closer to the maximum by taking steps. This is also commonly known as "iteratively reweighted least squares".

To find the confidence intervals of the coefficient estimates, we can use the `confint` function:

```
confint(logit.titanic)
```

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

```
##                   2.5 %     97.5 %
## (Intercept)  2.48429965  3.7676968
## pclass      -1.29190369 -0.9171451
## age         -0.04786159 -0.0264256
```

If we want to find a 90% confident interval for these input variables based on our logistical regression, we can specify that value as an argument to the function:

```
confint(logit.titanic, level=.90)
```

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

```
##                     5 %       95 %
## (Intercept)  2.58415204  3.6609399
## pclass      -1.26083805 -0.9464011
## age         -0.04609121 -0.0281061
```

2. The Likelihood Ratio Test

The Likelihood Ratio Test is used to compare 2 nested models. When you run a logistic regression, R can give you the model's log likelihood value using a function called ". Let's run a logistic regression model for which we're interested in the likelihood ratio test:

```
logit.titanic <- glm(survived ~ pclass + age,
                     family=binomial(link='logit'), data=titanic)
```

In the model above, the log likelihood is -628.09. You can display it by executing this command `logLik(glm.object)`:

```
logLik(logit.titanic)
```

```
## 'log Lik.' -628.0912 (df=3)
```

Using this value, you can calculate the likelihood ratio test statistic, as follows:

$$2 \times (\text{log likelihood}_{\text{full model}} - \text{log likelihood}_{\text{restricted model}})$$

where the restricted model is the "intercept-only model" where $\beta_1 = \beta_2 = 0$. This can be easily done by creating a new null `glm` model and following the above steps to find the log likelihood, and plugging into the equation for the likelihood ratio test:

```
logit.null.titanic <- glm(survived ~ 1,
                     family=binomial(link='logit'), data=titanic) # null model
```

You can compute the log-likelihood ratio test between a full and null model by hand above, or compute it in R with a function from the `lmtest` package called `lrtest` as shown below.

```
lrtest(logit.titanic)
```

```
## Likelihood ratio test
##
## Model 1: survived ~ pclass + age
## Model 2: survived ~ 1
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   3 -628.09
## 2   1 -707.31 -2 158.44  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What does this result tell you about the plausibility of the "full" model? In other words, what was your H_0 here, and how can you reject it?

3. Using `glht` to Find Estimates for Combinations of Coefficients

The `glht` function can be used to estimate lnear combinations of coefficients. This command is especially useful when your model contains interaction terms and you want to find confidence intervals around estimtaes of odds ratios that involve your interaction terms. For example:

```
glht.mod <- glht(logit.titanic)
confint(glht.mod)
```

```
##
##   Simultaneous Confidence Intervals
##
## Fit: glm(formula = survived ~ pclass + age, family = binomial(link = "logit"),
##     data = titanic)
##
## Quantile = 2.255
## 95% family-wise confidence level
##
##
## Linear Hypotheses:
##                   Estimate lwr      upr
## (Intercept) == 0  3.11432  2.37666  3.85198
## pclass == 0      -1.10146 -1.31689 -0.88604
## age == 0         -0.03699 -0.04931 -0.02467
```

4. The Predict Command

The `predict` command uses the estimated coefficients from the regression model that you most recently ran to assign a probability of disease (or whatever your outcome may be) to each observation in your dataset. These probabilities are contained in a newly created variable with whatever name you specify. For example, the following lines of code first fit a model (i.e., find the maximum likelihood coefficients), which you can then use to find the estimated probabilities of the outcome, survival, based on this model:

```
predicted.values <- predict(logit.titanic, type="response")
predicted.values <- cut(predicted.values, 2, labels=c(0, 1))
```

Here, we've scaled based on the response variable's range, so that anything above 0.5 is returned as a survived prediction of 1, and below or equal to 0.5 as 0. We can check our model's prediction accuracy by finding the equal categories between our original target variable `titanic$survived` and `predicted.values`. First, we must change the `titanic$survived` row into a factor vector because it is currently saved as an integer. Attempting to use equality operators between two classes can lead to skewed results.

```
sum(as.factor(titanic$survived) == predicted.values)/length(predicted.values)
```

```
## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```
## Warning in `==.default`(as.factor(titanic$survived), predicted.values):
## longer object length is not a multiple of shorter object length
```

```
## [1] 0.7045889
```

Our model's class prediction is 70.5% accurate.