Lab 7: Logistic Regression & GLM

PSC 103B

Marwin Carmo

Logistic regression

- Appropriate for binary outcomes, which only takes on values of 0 or 1.
- Oftentimes, 1 corresponds to a “success” of some type.
- Logistic regression predicts the log-odds of success because there is a linear relationship between the log-odds and your predictors.

Today’s dataset

- We are going to demonstrate logistic regression with this dataset on red wine quality, which can be found at https://archive.ics.uci.edu/ml/datasets/wine+quality.
- This dataset contains information on characteristics of the wine, such as the acidity, sugar, pH, alcohol content, etc. as well as a rating of the wine as “good” or “bad”.
- The rating was created by dichotomizing a quality variable (so that anything with a quality score greater than 5 out of 9 was considered good).

Today’s dataset

```r
dplyr::glimpse(wine$quality)
```

```
chr [1:1599] "bad" "bad" "bad" "good" "bad" "bad" "bad" "good" "good" ...
```

- We are going to create a second column that assigns this a numerical value (1 if the quality is good, 0 if it’s bad).
- That way, we can be sure our logistic regression is predicting the log-odds of a wine being good.

```r
wine$quality_binary <- ifelse(wine$quality == "good", 1, 0)

dplyr::glimpse(wine$quality_binary)
```

```
num [1:1599] 0 0 0 1 0 0 0 1 1 0 ...
```

Simple Logisitic Regression

- Let's fit a logistic regression model with only one predictor – alcohol content.
- To fit a logistic regression function, we need to use a new function called `glm()`.
- This function, for the most part, is similar to the `lm()`: `outcome var ~ independent var(s)`.
- We just have to specify a family, using `family = "binomial"` (this tells `R` that the outcome variable is binary, and to use logistic regression).

```
simple_logreg <- glm(quality_binary ~ alcohol,
                     data = wine,
                     family = "binomial")
```

```r
summary(simple_logreg)
```

```
Call:
glm(formula = quality_binary ~ alcohol, family = "binomial",
    data = wine)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -10.76302    0.68326  -15.75   <2e-16 ***
alcohol       1.05559    0.06663   15.84   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2209  on 1598  degrees of freedom
Residual deviance: 1865  on 1597  degrees of freedom
```

Simple Logisitic Regression

$$\log(Odds_{\text{Quality}}) = -10.76 + 1.06 \times \text{Alcohol}$$

- The **intercept** of -10.76 means that a red wine that has 0 alcohol content has an expected log-odds score of -10.76.
- The **slope** means that for every 1-unit increase in alcohol content, the log-odds of a wine being rated good increase by 1.06 points.
- But how do we interpret a log-odds?
- It is not intuitive, and that's why we transform the coefficients to be interpreted in terms of odds ratios. How can we do that?

Simple Logisitic Regression

$$Odds_{\text{Good}} = \exp(-10.76 + 1.06 \times \text{Alcohol})$$

or (using the rules of exponents),

$$Odds_{\text{Good}} = \exp(-10.76) \times \exp(1.06 \times \text{Alcohol})$$

Interpreting the coefficients

- **Intercept**: it is still the expected value when alcohol is 0. So when a wine has no alcohol content, the expected odds of being rated good are exp(-10.76), or .00002.
- **Slope**: it is the multiplicative change in the odds for a 1-unit change in Alcohol. So when alcohol content increases by 1, the odds increase by a factor of (are multiplied by) $e^{1.06}$ = 2.89, or a 289% increase in the odds.
- Because 1.06 is positive, we know that as the alcohol content increases, the probability of a wine being considered good also increase.

Mean centering the predictors

```r
wine$alcohol_c <- wine$alcohol - mean(wine$alcohol, na.rm = TRUE)
```

```r
logreg_centered <- glm(quality_binary ~ alcohol_c,
                       data = wine,
                       family = "binomial")

summary(logreg_centered)
```

```
Call:
glm(formula = quality_binary ~ alcohol_c, family = "binomial",
    data = wine)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.23937    0.05753    4.16 3.18e-05 ***
alcohol_c    1.05559    0.06663   15.84  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2209  on 1598  degrees of freedom
```

Mean centering the predictors

- Notice that the value of the slope hasn't changed, but now our intercept is 0.24 – what does this mean?
- The expected log-odds of a wine with an average alcohol content being rated good is 0.24.
- The odds of it being rated good are $e^{0.24}$ = 1.27 so it's more likely to be rated good than it is to be rated bad.

Predicted probabilities

- What is the probability that the wine will be rated “good”?
- Let’s take the example of a Cabernet Sauvignon, which generally has an alcohol content of 14%.

```r
(logodds <- -10.76 + 1.06*14)
```

```
[1] 4.08
```

- The expected log-odds are 4.08, which again, are hard for us to interpret so let’s transform those into odds.

```r
(odds <- exp(logodds))
```

```
[1] 59.14547
```

Predicted probabilities

- We can transform these odds into a probability using, $P = \frac{odds}{1+odds}$

```
odds / (1 + odds)
```

```
[1] 0.9833736
```

- A wine with 14% alcohol has almost 100% (98.34%) chance of being rated good according to our model.

Predicted probabilities

- A probability of 0.5 (equal chance) corresponds to an odds ratio of 1, which corresponds to a log-odds of 0.
- Therefore, if something has a log-odds greater than 0, then that **increases** your chance of a success (probability greater than 0.5), and a log-odds less than 0 **decreases** your chance of success (probability less than 0.5).

Now you try

Exercise

Fit a model predicting quality from the amount of chlorides, which can affect how salty a wine tastes. Interpret the coefficients in terms of log-odds and odds.

Multiple Logistic Regression

- Just like in linear regression, we can add multiple predictors to our logistic regression model.

```r
multiple_logreg <- glm(quality_binary ~ alcohol + sulphates,
                       data = wine,
                       family = "binomial")
summary(multiple_logreg)
```

```
Call:
glm(formula = quality_binary ~ alcohol + sulphates, family = "binomial",
    data = wine)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -12.32438    0.73149 -16.848  < 2e-16 ***
alcohol       1.03724    0.06728  15.417  < 2e-16 ***
sulphates     2.67209    0.37097   7.203 5.89e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

Multiple Logistic Regression

$$\log(Odds_{\text{Good}}) = -12.32 + 1.04 \times \text{Alcohol} + 2.67 \times \text{Sulphates}$$

- **Intercept**: The expected odds of a wine with no alcohol content and no sulphates is exp(-12.32) = .000004.
- **Slope of Alcohol**: Holding the amount of sulphates constant, increasing the alcohol content by 1 increases the odds of a wine being rated as good by a factor of exp(1.04) = 2.83.
- **Slope of Sulphates**: Holding the amount of alcohol constant, increasing the sulphates by 1 unit increases the odds of a wine being rated as good by a factor of exp(2.67) = 14.44.

Multiple Logistic Regression with interactions

```r
interact_logreg <- glm(quality_binary ~ alcohol + sulphates + I(alcohol*sulphates),
                      data = wine,
                      family = "binomial")
summary(interact_logreg)
```

```
Call:
glm(formula = quality_binary ~ alcohol + sulphates + I(alcohol *
    sulphates), family = "binomial", data = wine)

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)             1.8977     3.0787   0.616    0.538
alcohol                -0.3986     0.3104  -1.284    0.199
sulphates             -19.7478     4.8400  -4.080 4.50e-05 ***
I(alcohol * sulphates)   2.2643     0.4897   4.624 3.77e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

General Linear Model

General Linear Model

- Many of the tests we learned in class can be rewritten as linear regressions with the help of dummy coding.
- Let’s revisit our ANOVA example from a few weeks ago where we wanted to examine bill length differences between different species of penguins.
- We had to do an F-test to determine whether any of the means were different, and then a post-hoc test to see which means were different.

General Linear Model

- We can fit this model as a regression model with dummy codes and this might help us make more specific comparisons right away.
- A dummy code is a variable that assigns a value of 1 if a person is in one specific group, and 0 otherwise.
- You can manually create these dummy codes, or if you fit a regression model with a “factor” variable as a predictor, R will automatically create dummy codes.

General Linear Model

```r
# Load the palmerpenguins package
library(palmerpenguins)
```

```r
dummyreg <- lm(bill_length_mm ~ species, data = penguins)
summary(dummyreg)
```

```
Call:
lm(formula = bill_length_mm ~ species, data = penguins)

Residuals:
    Min      1Q  Median      3Q     Max 
-7.9338 -2.2049  0.0086  2.0662 12.0951 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)       38.7914     0.2409  161.05   <2e-16 ***
speciesChinstrap  10.0424     0.4323   23.23   <2e-16 ***
speciesGentoo      8.7135     0.3595   24.24   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

General Linear Model

- R automatically assigned one group to be our reference group and that’s generally done by choosing the group that comes first alphabetically (in this case, Adelie).
- This reference group has a score of 0 on the dummy code variables.
- Now the slopes are the effects of the dummy codes for the other 2 species.

$$y_i = b_0 + b_1\text{Chinstrap} + b_2\text{Gentoo}$$

- If a penguin is from the Adelie species:

$$y_i = b_0 + b_1(0) + b_2(0)$$

- If a penguin is from the Chinstrap species:

$$y_i = b_0 + b_1(1) + b_2(0)$$

- If a penguin is from the Gentoo species:

$$y_i = b_0 + b_1(1) + b_2(1)$$

- The intercept now represents the expected bill length for a penguin that has a 0 on both dummy codes. That is, 38.79 is the average bill length for Adelie penguins.
- The slopes represent the difference between the mean of the reference group, and the mean of the group that the dummy code is for:
  - Chinstrap penguins have an average bill length 10.04 mm longer than the average bill length of Adelie penguins.
  - Gentoo penguins have an average bill length 8.71 mm longer than the average bill length of Adelie penguins.

General Linear Model: coefficients

- Both these differences are significant! So unlike before where we had to do a post-hoc test to see which groups were different, we can automatically see which pairs are significantly different.
- However, not all comparisons are represented - we can’t say anything about the difference between Chinstrap and Gentoo penguins.