Lab 05 - Post-hoc Tests & Factorial ANOVA

PSC-103B

Marwin Carmo

2025-02-13

Today, we will be learning about logistic regression which is when you want to fit a regression model to some binary data, which only takes on values of 0 or 1. Remember that oftentimes, 1 corresponds to a "success" of some type, whether that is getting into graduate school, or receiving a promotion, or flipping a heads on a coin (it's all about how you want to define success).

Since our outcome is binary, we can't use our traditional linear regression model to predict a success or failure. This is because we're trying to predict a probability of success, and probability is bounded between 0 and 1. Linear regression assumes your outcome is continuous, or unbounded, so we need a model that respects the fact that probability is bounded.

Enter logistic regression! But as we saw in class, logistic regression doesn't predict probability directly because the regression equation would be complicated to estimate. Instead, logistic regression predicts the log-odds of success because there is a linear relationship between the log-odds and your predictors, so we can use OLS regression to estimate the model (which is simpler!)

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", which was created by dichotomizing a quality variable (so that anything with a quality score greater than 5 out of 9 was considered good).

Let's load it in.

```
wine <- read.csv("https://shorturl.at/Vwcr6")

head(wine)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
## 4                  17                   60  0.9980 3.16      0.58     9.8
## 5                  11                   34  0.9978 3.51      0.56     9.4
## 6                  13                   40  0.9978 3.51      0.56     9.4
##   quality
## 1     bad
```

```
## 2     bad
## 3     bad
## 4    good
## 5     bad
## 6     bad
```

Notice that right now, the quality variable has two options bad or good; to make this easier for us, 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.

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

Did this work?

```
head(wine)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
## 4                  17                   60  0.9980 3.16      0.58     9.8
## 5                  11                   34  0.9978 3.51      0.56     9.4
## 6                  13                   40  0.9978 3.51      0.56     9.4
##   quality quality_binary
## 1     bad              0
## 2     bad              0
## 3     bad              0
## 4    good              1
## 5     bad              0
## 6     bad              0
```

Simple Logisitic Regression

To start, let's fit a logistic regression model with only one predictor - what if we thought that alcohol content was a good indicator of whether or not a red wine was considered good or bad?

You might remember that we fit linear regression models using the `lm()` function. However, since we need 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()` function: you specify a formula in the form `outcome var ~ independent var(s)` and then you provide some data. The only extra thing you have to do now is specify a family, using `family = "binomial"` (this tells `R` that the outcome variable is binary, and to use logistic regression).

Let's fit the model:

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

Just like in linear regression, we can access information about the model using the `summary()` function:

```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
## AIC: 1869
##
## Number of Fisher Scoring iterations: 4
```

Let's write out this model and interpret it

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

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.

Hopefully this interpretation seems pretty similar to how we interpreted things in linear regression. Only now our outcome variable is the log-odds of something being successful (here, the log-odds of a wine being rated good) and that's a little bit difficult for people to interpret because it's hard for us to have an intuitive sense of these things.

So often, people will transform the coefficients to be interpreted in terms of odds ratios. And you can do that by simply exponentiating!

If we were to re-write this model in terms of the odds we would have:

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

or (using the rules of exponents),

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

So what do these values mean now?

Well, our intercept 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 (these odds are super tiny – people don't like wine with no alcohol!). Now, the slope is exp(1.06), and 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) exp(1.06) = 2.89. This is almost a 300% 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.

Just like in linear regression, we can mean-center our predictor variables to improve the interpretation of our intercept – because a wine with an alcohol content of 0 is pretty unlikely (unless its a special, NA wine).

Let's create a centered version of alcohol:

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

And use that as a predictor,

```
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
## Residual deviance: 1865  on 1597  degrees of freedom
## AIC: 1869
##
## Number of Fisher Scoring iterations: 4
```

Notice that the value of the slope hasn't changed, but now our intercept is 0.24 – what does this mean? It means the expected log-odds of a wine with an average alcohol content being rated good is 0.24 or that the odds of it being rated good are exp(.24) = 1.27 so it's more likely to be rated good than it is to be rated bad.

One other thing you can look at with the output of a logistic regression is the predicted probability - for a particular alcohol content, 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%.

What are the predicted log-odds of that wine being rated "good"?

```
logodds <- -10.76 + 1.06*14
logodds
```

```
## [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.

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

```
## [1] 59.14547
```

the expected odds of a cab being rated good is **huge**, 59.15. A cab is 59 times as likely to be rated good as it is to be rated bad.

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

```
prob <- odds / (1 + odds)
prob
```

```
## [1] 0.9833736
```

A cab has almost 100% (98.34%) chance of being rated good according to our model.

Some rules that might be helpful to know is that 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 – fit a model predicting quality from the amount of chlorides, which can affect how salty a wine tastes. And interpret the coefficients in terms of log-odds and odds.

```
log_fit <- glm(quality_binary ~ chlorides,
               data = wine,
               family = "binomial")
summary(log_fit)
```

```
##
## Call:
## glm(formula = quality_binary ~ chlorides, family = "binomial",
##     data = wine)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.6054     0.1241   4.879 1.07e-06 ***
## chlorides    -5.3767     1.3284  -4.047 5.18e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2209.0  on 1598  degrees of freedom
## Residual deviance: 2188.6  on 1597  degrees of freedom
## AIC: 2192.6
##
## Number of Fisher Scoring iterations: 3
```

Multiple Logistic Regression

Just like linear regression had a "simple" and "multiple" version based on the number of predictors, we can add multiple predictors to our logistic regression model. Let's include alcohol content in our logistic regression model, like before, but now add sulphates.

```
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)
##
##     Null deviance: 2209.0  on 1598  degrees of freedom
## Residual deviance: 1804.7  on 1596  degrees of freedom
## AIC: 1810.7
##
## Number of Fisher Scoring iterations: 4
```

What is our regression model? And what do the values mean?

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

- Intercept: A wine with no alcohol content and no sulphates has a log-odds of -12.32.
- Slope of Alcohol: For each 1-unit increase in alcohol content, holding amount of sulphates constant, the log-odds of a wine being rated good increase by 1.03.
- Slope of Sulphates: For each 1-unit increase in sulphates, holding alcohol content constant, the log-odds increase by 2.67.

But let's interpret these in terms of the odds

- Intercept: The expected odds of a wine with no alcohol content and no sulphates is exp(-12.32) = .000004 (again, almost 0!)
- 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. The odds are multiplied by 2.83 for each 1-unit increase in alcohol content, holding the amount of sulphates constant **or** The odds increase by 183% for each 1-unit increase in alcohol content, holding the amount of sulphates constant.
- 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. The odds are multiplied by 14.44 for each 1-unit increase in sulphate content, holding the amount of alcohol constant **or** The odds increase by 1343% for each 1-unit increase in sulphate content, holding the amount of alcohol constant.

Of course, you can also add interactions to your logistic regression model.

```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)
##
##     Null deviance: 2209.0  on 1598  degrees of freedom
## Residual deviance: 1782.2  on 1595  degrees of freedom
## AIC: 1790.2
##
## Number of Fisher Scoring iterations: 5
```

We're not going to interpret the values of this model, other than to say that the significant interaction means that the effect of alcohol on the log-odds (or odds, or probability) of a wine being rated good depends on the amount of sulphates (or vice versa). What is that dependence? You'd have to plot the relation between alcohol and the log-odds for a few different values of sulphates to find out.

General Linear Model

One final thing you all learned about was the general linear model, and how a lot of the tests you guys have learned about in class – t-tests and ANOVA – 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. And 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.

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

But we can fit this as a regression model with dummy codes and this might help us make more specific comparisons right away, particularly if there was one single group like a control group, that we were interested in comparing our other groups to.

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.

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

```
##
## 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
##
```

```
## Residual standard error: 2.96 on 339 degrees of freedom
##   (2 observations deleted due to missingness)
## Multiple R-squared:  0.7078, Adjusted R-squared:  0.7061
## F-statistic: 410.6 on 2 and 339 DF,  p-value: < 2.2e-16
```

So what does this output mean? Well, `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. And now the slopes are the effects of the dummy codes for the other 2 species.

So our intercept now represents the expected bill length for a penguin that has a 0 on both dummy codes – in other words, a penguin in the reference group, Adelie! The value of 38.79 is the average bill length for Adelie penguins. What do the slopes represent? The difference between the mean of the reference group, and the mean of the group that the dummy code is for.

In other words, the slope of 10.04 for `speciesChinstrap` means that Chinstrap penguins have an average bill length that is 10.04 mm longer than the average bill length of Adelie penguins. And the value of 8.71 for `speciesGentoo` means that Gentoo penguins have an average bill length that is 8.71 mm longer than the average bill length of Adelie penguins.

We can see if this matches:

```
tapply(penguins$bill_length_mm, penguins$species, mean,
       na.rm = TRUE)
```

```
##    Adelie Chinstrap    Gentoo
##  38.79139  48.83382  47.50488
```

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.

Of course, the only downside to this approach is that not all comparisons are represented - we can't say anything about the difference between Chinstrap and Gentoo penguins. That's why this approach is often performed for comparisons among some type of experimental group and a clear control group!