Linear & Logistic Regression Materials

Brier Gallihugh, M.S.

2024-02-23

Table of contents

An Introduction to Regression

Linear Regression

```
library(tidyverse)
library(car)

data <- starwars %>% select(height,mass,sex) %>% na.omit()                    ①
```

① The above code takes the `starwars` data set and uses the `select()` function to pull out the columns labelled height, mass and sex. Then using the `na.omit()` function, I've removed (using listwise deletion) any rows which contain missing values

Running a Linear Regression

Similar to correlation, regression (particularly multiple regression) is very common in the social sciences. As such, lets dive into an example again using the `starwars` data set.

```
data <- data %>%
  filter(sex == "male" | sex == "female") %>%                    ①
  mutate(sex = dplyr::recode(sex,                                ②
                             "male" = 1,
                             "female" = 0))

linear_regression <- lm(height ~ sex + mass, data = data)        ③

summary(linear_regression)                                       ④
```

① Here I want to filter out the data set for observations (rows) that meet the following conditions (in this case those with a sex "value" of either "male" or "female").
② I then want to recode the sex variables to a numeric value for the purposes of doing the linear regression using the `recode()` function. It takes the column as an input and takes the syntax "old value" to "new value".
③ This is a basic linear regression formula using the `lm()` function. It takes the form DV ~ IV + IV, df).
④ To see the results of the regression, we simply run the `summary()` function and specify the name we assigned the regression (i.e., linear_regression).

```
Call:
lm(formula = height ~ sex + mass, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-47.381  -4.146   1.359   9.036  48.522

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 115.1002     9.8771  11.653 1.34e-15 ***
sex         -20.1723     8.6881  -2.322   0.0245 *
mass          1.0323     0.1201   8.596 2.81e-11 ***
```

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

Residual standard error: 22.13 on 48 degrees of freedom
Multiple R-squared:  0.6081,    Adjusted R-squared:  0.5917
F-statistic: 37.23 on 2 and 48 DF,  p-value: 1.727e-10
```

Assumptions of Linear Regression

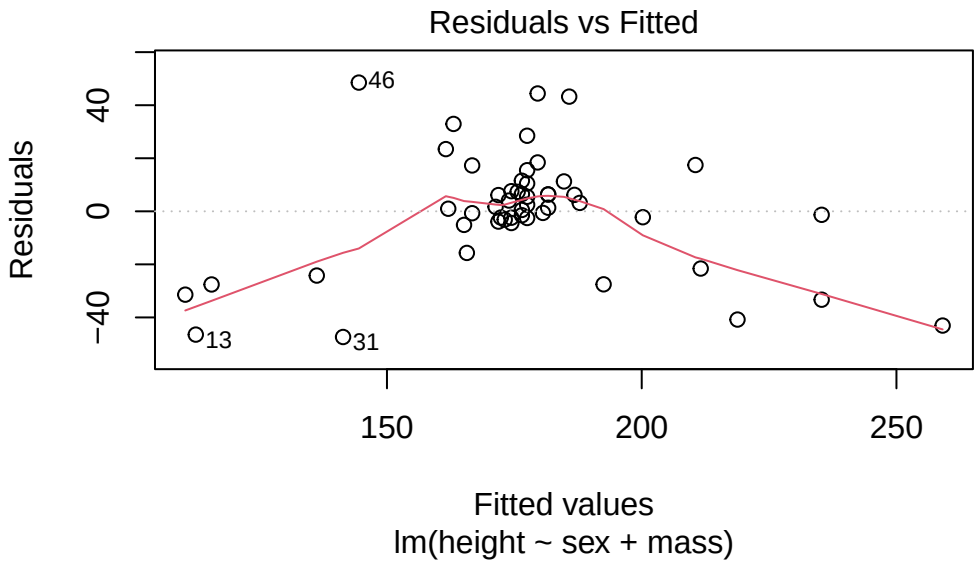
As some will attest to, we as social scientists don't always explicitly test our statistical assumptions (this is a problem). However, as a parametric test, regression consists of the following key assumptions: homogeneity of variance, residual normality, lack of multicollinearity, and the independence of errors. Further, we likely want to at least investigate the potentiality that there are outliers and influential cases. I'm going to show you how to test each of these assumptions.

Homogeneity of Variance

Graphical

```r
# Should Be a Straight Line
plot(linear_regression,1)                                                    ①
```

① The `plot(object,1)` function when applied to a regression object will give you a plot assessing visually the homogeneity of variance assumption. The line should be roughly straight.



Statistical

```
library(lmtest)
# Goldfield Quandt Test (Less than .05 BAD)
gqtest(linear_regression)                                    ①
```

① Statistically, we can also investigate homogeneity of variance using a Goldfield Quandt Test. This is one test we hope is not statistically significant. To do this, we use the `gqtest()` function in the `lmtest` package.

```
    Goldfeld-Quandt test

data:  linear_regression
GQ = 1.4761, df1 = 23, df2 = 22, p-value = 0.1825
alternative hypothesis: variance increases from segment 1 to 2
```

Residual Normality

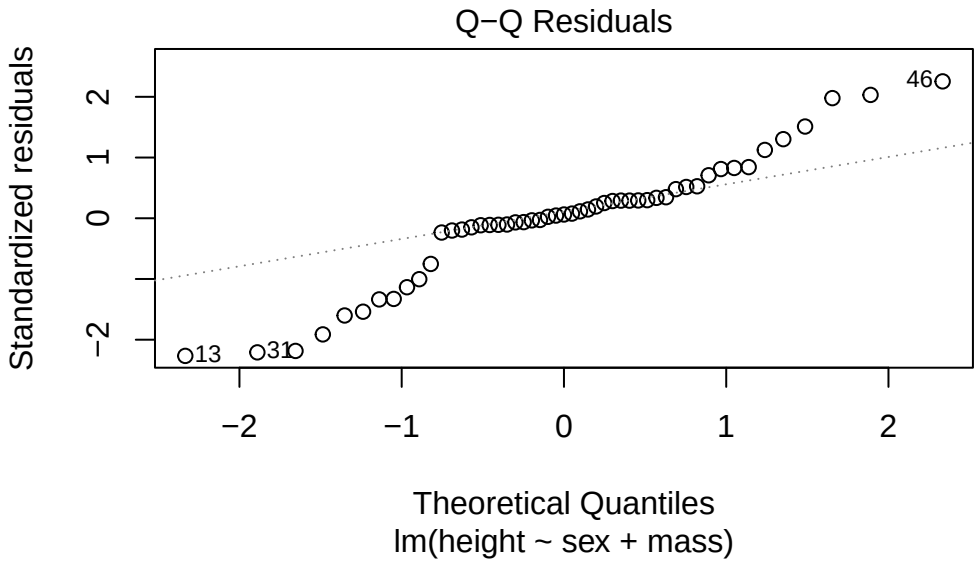
Contrary to popular belief, normality doesn't typically refer to the distribution of each individual variable in a regression. What matters is that the model residuals will be roughly normally

distributed. Below we will go through this. First, we'll look at the assumption graphically using what is known as a qq plot

Graphical

```
# Should Follow Straight Diagonal Line
plot(linear_regression,2)                                    ①
```

① Use the `plot(object,2)` with a regression will give you a standard qqplot with a reference line. This line should look linear.



Statistical

We can also assess the assumption statistically using a Shapiro Wilks test. Keep in mind this test can be heavily influenced by sample size but nonetheless it is a start.

```
shapiro.test(residuals(linear_regression))                   ①
```

① The `shapiro.test()` function will get us what we want. However, we need to make sure we get the residuals of the model so we need to use the `residuals()` function too.

```
    Shapiro-Wilk normality test

data:  residuals(linear_regression)
W = 0.94273, p-value = 0.01582
```

Multicollinearity

Multicollinearity (or too high of correlation between variables) can be an issue. One could look at the correlation matrix but that's highly subjective. A better idea might be to use what is known as variance inflation factors. Essentially higher values (i.e., north of 10) indicate an issue with said factor (i.e., variable). Below is the code for this using the `car` package.

Statistical (VIF)

```
# Individual Predictors (Less than 10 is OK)
car::vif(linear_regression)                                    ①

# Mean Across Predictors (Ideally Around 1)
mean(vif(linear_regression))                                   ②

# Tolerance (Greater than .20 Ideal)
1/vif(linear_regression)                                       ③
```

① Less than 10 is solid for this metric
② You want the mean value to be around 1.0
③ You want tolerance to be $>$ .20

```
    sex    mass
1.14225 1.14225
[1] 1.14225
      sex      mass
0.8754655 0.8754655
```

Independence of Errors

Long story short, errors shouldn't be correlated with each other. We can assess this statistically using the Durbin Watson test. Here we want the test to not be statistically significant. We can use the `durbinWatsonTest()` function from the `car` function for this. The code is shown below.

Statistical (Durbin Watson Test)

```
library(car)
car::durbinWatsonTest(linear_regression)
```

```
 lag Autocorrelation D-W Statistic p-value
   1       0.3639846      1.237263   0.004
 Alternative hypothesis: rho != 0
```

Finding Outliers & Influential Cases

Residuals

Typically, residuals outside of the range of -2.5 to 2.5 are a potential problem. However, this doesn't mean we should discard the data necessarily. We have to feel they are distinctly not in the population we're hoping to investigate. However, typically as a rule of thumb, having more than 5% of your cases being outside of this range is not ideal.

```
# Greater than 5% of Data Potentially Problematic
data$residuals <- resid(linear_regression)
summary(data$residuals)

problem_residuals <- data %>% filter(residuals > 2.5 | residuals < -2.5)     ①

nrow(problem_residuals)/nrow(data)                                           ②
```

① This will filter the data set into observations which meet the criteria
② This tells me what percent of the cases are potentially problematic

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-47.381  -4.146   1.359   0.000   9.036  48.522
[1] 0.7647059
```

Influential Cases

Maybe a more useful investigation is that involving influential cases (i.e., cases which have undue influence on the results). There are several metrics one can you. I'm just going to focus on Cooks distance. Essentially values over 1 are potentially problematic. Below is the code for this

```r
data$cooks_dist <- round(cooks.distance(linear_regression),5)                    ①
# Greater Than 1 is Problematic
summary(data$cooks_dist)                                                         ②
```

① This rounds the values to 5 decimal places for ease of reading in addition to calculating the Cooks distance for each observation.

② `summary()` function says the max value is .36 so there are no individual cases here that are having undue influence on the results.

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.000000 0.000185 0.001580 0.033735 0.030940 0.414110
```

Logistic Regression

So logistic regression is something I had to learn in order to do this workshop. Essentially however, logistic regression is used to give odds ratios of whether an observation belongs in one of two categories based on a set of inputs. In industry, this is actually considered a form of machine learning (so is most other regression). You can actually go above and beyond two categories but we're not going to talk about polynomial regression in this workshop. Below is how we run a logistic regression in R.

Running a Logistic Regression

```r
# Create Logistic Model w/ Multiple Predictors
log_regression <- glm(sex ~ height + mass, family = binomial(link = "logit"), data = data) ①
summary(log_regression)                                                          ②

# Determine Chi Square Diff
modelChi <- log_regression$null.deviance - log_regression$deviance               ③
modeldf <- log_regression$df.null - log_regression$df.residual
chisq_prob <- 1 - pchisq(modelChi,modeldf)
print(chisq_prob)

# Odds Ratios
exp(log_regression$coefficients)                                                 ④

# CI
exp(confint(log_regression))                                                     ⑤
```

```
# Effect Size
effect_size <- modelChi/log_regression$null.deviance                    ⑥
print(effect_size)
```

① This is your basic logistic regression formula. The `family = binomial(link = "logit")` tells R that we want this model to be a logistic regression.
② Summarizes the results of the logistic regression
③ Calculates a Chi Square Test to determine if the model is better than chance (less than .05)
④ Exponentiation the coefficients will give you the odds ratios
⑤ We can also get the confidence intervals of the odds ratios using the `confint()` function
⑥ We can calculate an effect size using this formula

```
Call:
glm(formula = sex ~ height + mass, family = binomial(link = "logit"),
    data = data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.29283    2.10063   1.568  0.11699
height      -0.06352    0.02392  -2.655  0.00793 **
mass         0.14111    0.04618   3.056  0.00224 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 47.532  on 50  degrees of freedom
Residual deviance: 30.172  on 48  degrees of freedom
AIC: 36.172

Number of Fisher Scoring iterations: 6

[1] 0.0001699965
(Intercept)      height        mass
 26.9188822   0.9384563   1.1515534
                2.5 %       97.5 %
(Intercept) 0.5828766 3129.1281391
height      0.8879965    0.9781015
mass        1.0662872    1.2866909
[1] 0.365217
```

Assumptions of Logistic Regression

Like regular regression, logistic regression also has assumptions that must be met. They are the linearity of each predictor with the **log** of the outcome, independence of errors, and multicollinearity. I will show you how to test each of these below.

Linearity w/ Log of Outcome (Each Predictor)

Statistical

```
data$massINT <- data$mass*log(data$mass)                                    ①
data$heightINT <- data$height*log(data$height)

linearity_assumption <- glm(sex ~ height + mass + massINT + heightINT, family = binomial(lin

summary(linearity_assumption)
```

① You want the interaction terms of each predictor with the log of the outcome. These need to go into the logistic regression.
② You can see them added here. You want none of the interaction terms to be statistically significant

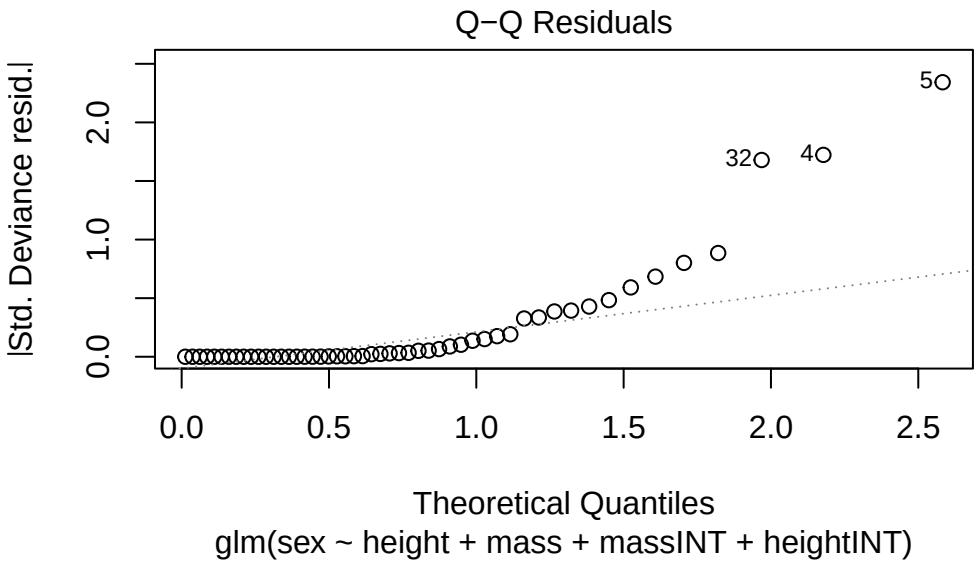
```
Call:
glm(formula = sex ~ height + mass + massINT + heightINT, family = binomial(link = "logit"),
    data = data)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) 1045.1876   700.2203   1.493   0.1355
height       -41.4124    26.7390  -1.549   0.1214
mass           5.4306     3.1538   1.722   0.0851 .
massINT       -0.9786     0.5835  -1.677   0.0936 .
heightINT      6.7654     4.3627   1.551   0.1210
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 47.5319  on 50  degrees of freedom
Residual deviance:  8.5006  on 46  degrees of freedom
AIC: 18.501
```

```
Number of Fisher Scoring iterations: 13
```

Graphical

You can also just plot the logistic regression with the interaction terms and see if you get a roughly straight line.

```
plot(linearity_assumption,2)
```



Independence of Errors

Like regular regression, we can test this assumption using a Durbin Watson Test. The code here is below.

Statistical

```
durbinWatsonTest(log_regression)
```

```
 lag Autocorrelation D-W Statistic p-value
   1      0.02164757      1.940752   0.814
 Alternative hypothesis: rho != 0
```

Multicollinearity

Multicollinearity can also be assessed just like regression by using the `vif()` function as well as the `1/vif()` function. We want these to be less than 10 and greater than .20 respectively.

Statistical

```
vif(log_regression)
```

```
  height     mass
4.517338 4.517338
```

```
1/vif(log_regression)
```

```
   height      mass
0.2213693 0.2213693
```

Finding Outliers & Influential Cases

Residuals

Residuals can be assessed the same was as it is in regular linear regression. The code is below as a refresher.

```
# Greater than 5% of Data Potentially Problematic
data$residuals_log <- resid(log_regression)
summary(data$residuals_log)
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-2.61860  0.03527  0.28204  0.11555  0.38815  2.14419
```

```
log_prob <- data %>% filter(residuals_log > 2 | residuals_log < -2)

round(nrow(log_prob)/nrow(data),3)
```

```
[1] 0.039
```

Influential Cases

As with residuals and linear regression compared to logistic regression, the same is true for influential cases. You can see the code for this below.

```
data$cooks_dist_log <- cooks.distance(log_regression)
# Greater Than 1 is Problematic
summary(data$cooks_dist_log)
```

```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.0000000 0.0003326 0.0007367 0.0267416 0.0233058 0.4964929
```