STA 214 Homework 5

Due: Friday, February 17, 12:00pm (noon) on Canvas.

Instructions: There are three parts to this assignment. Part 1 is practice with maximum likelihood estimation, Part 2 is a simulation study exploring the effects of outliers on logistic regression models, and Part 3 is a short (extra credit) problem on debugging in R.

Getting started: Begin by downloading the HW5 template from the course website:

`https://sta214-s23.github.io/homework/hw_05_template.Rmd`

Save this template file to your computer, then open it in RStudio. As you complete the assignment, you will write down your answers to all questions in the R Markdown file, and include all R code in code chunks. *If a question requires code, you will not receive credit if no code is provided.* Refer to the R Markdown instructions on the course website (`https://sta214-s23.github.io/resources/rmarkdown_instructions/`) if you have issues getting started.

Submission: When you have completed the assignment, knit your homework to HTML and submit on Canvas.

1 Maximum likelihood estimation

This part focuses on practice with maximum likelihood estimation. A brief review of mathematical notation and rules for logs and derivatives can be found in the course codebook.

Suppose we have a random variable Y, which can take on values $y = 0, 1, 2, 3, 4, ...$ (i.e., any non-negative integer). The probability of each outcome does not depend on any explanatory variables, and we are told that

$$P(Y = y) = \frac{(y + \theta - 1)!}{y!(\theta - 1)!}(1 - p)^{\theta} p^{y}$$

where θ is known, but p is *unknown*.

1. We want to estimate p, so we take a sample of data $Y_1, ..., Y_n$ and we want to calculate the maximum likelihood estimate $\widehat{p}$ as a general function of $Y_1, ..., Y_n$ and θ. The work for this problem will feel similar to question 2 on HW 4: the same rules for working with sums and products apply, we just have a different distribution.

   (a) Calculate the log likelihood $\ell(p)$ when we observe data $Y_1, ..., Y_n$ and $P(Y = y) = \frac{(y + \theta - 1)!}{y!(\theta - 1)!}(1 - p)^{\theta} p^{y}$. Your answer should involve the Y_is, and should involve a sum.

   (b) Using the log likelihood from part (a), find the MLE $\widehat{p}$. Your answer should be in terms of the Y_is and θ.

2 A small simulation study

Simulation allows us to explore the behavior of models, confidence intervals, and hypothesis tests under certain conditions. One example of simulation is generating an approximate null distribution in parametric bootstrapping – in other words, how the test statistic would behave *if* the null hypothesis were true. Other examples of simulation are the class activities in which we simulated data with assumption violations to see how our fitted models and diagnostics would behave.

The purpose of this section is to use simulation to explore the behavior of *confidence intervals*. We want to know how confidence intervals behave when regression assumptions are violated. We will answer this question by simulating many sets of data, calculating confidence intervals, and seeing how often our confidence intervals actually capture the true parameter of interest.

2. Before we explore the behavior of confidence intervals when regression assumptions are violated, let's explore their behavior when the assumptions are all satisfied.

   (a) Adapt code from class (e.g., the class activity on February 8) to simulate a dataset $(X_1, Y_1), ..., (X_{100}, Y_{100})$ from the following model:

   $$\begin{aligned} X_i &\sim N(0,1) \\ \pi_i &= \frac{\exp\{-2+3X_i\}}{1+\exp\{-2+3X_i\}} \\ Y_i &\sim Bernoulli(\pi_i) \end{aligned}$$

   (b) Fit a logistic regression model on the simulated data, and create a 95% confidence interval for β_1 ($\beta_1 = 3$ in your simulation). Does your interval capture the true β_1?

   (c) Explain what it means for your interval to be a "95% confidence interval".

   (d) Use a `for` loop to repeat (a) and (b) 1000 times. What fraction of your intervals do you expect to contain β_1? What fraction actually contain β_1? *Hints:*

   - You will need a way to programatically calculate the upper and lower endpoints of the interval at each iteration. If `m1` is your fitted model, then `summary(m1)$coefficients` will produce a matrix of the estimated coefficients, their standard errors, and the test statistics. The estimate $\widehat{\beta_1}$ can be accessed by `summary(m1)$coefficients[2,1]`, and the standard error of $\widehat{\beta_1}$ can be accessed by `summary(m1)$coefficients[2,2]`
   - You will need a way to check whether your endpoints at each iteration capture the true β_1. If the upper endpoint is called `upper` and the lower endpoint is called `lower`, then

     ```
     upper > 3 && lower < 3
     ```

     will check whether the endpoints contain $\beta_1 = 3$.

   (e) Modify (d) to construct 99% confidence intervals instead, and verify that your intervals do indeed have the desired coverage probability.

3. Now let's add an outlier to the data, and see how it changes the coverage probability of our confidence intervals.

   (a) Modify your code from 2(d) by adding an outlier at $x = -2$ and $y = 1$ (see the class activity from February 8). Use the data with the outlier to fit the logistic regression model at each iteration. How does the outlier change the coverage probability of your confidence intervals?

(b) Repeat 3(a) for outliers at each of the following locations, calculating a coverage probability for each outlier location:

- $x = -1, y = 1$
- $x = -3, y = 1$
- $x = -4, y = 1$
- $x = -5, y = 1$

Then create a plot showing coverage probability on the vertical axis, and the x coordinate of the outlier on the horizontal axis. What do you conclude about outliers and coverage probabilities?

3 Extra credit: Debugging practice

This part is separate from the previous parts. The purpose of this section is to practice debugging the errors that we sometimes encounter in R and RStudio. This part is *optional*, and a correct submission will earn a small number of extra credit points on the assignment.

4. Your friend is working with the titanic data, and wants to model the relationship between fare and the survival. After doing some exploratory data analysis, they decide that a square transformation on fare is important. So, they want to create a new variable in the titanic data called `sqrtFare`. They write the following code in R:

```
library(tidyverse)

titanic <- read.csv("https://sta214-s23.github.io/homework/Titanic.csv")
titanic <- titanic %>%
  drop_na()

titanic %>%
  mutate(sqrtFare = sqrt(Fare))

m1 <- glm(Survived ~ sqrtFare, data = titanic, family = binomial)
```

However, when they run their code, they get an error!

(a) What error appears when you run this code? What is causing the error message?

(b) Explain to your friend how they can fix the error in their code.