Week 3
Tutorial - MLE

Applied Statistical Analysis II

Lecturer: Jeffrey Ziegler, PhD
Teaching Fellow: Shekhar Kedia

Spring 2026

Roadmap for today

- Today:
  - MLE
    1. Know how to create own likelihood function
    2. Run your own GLM with MLE
    3. Feel comfortable with what's happening when you execute glm()
- By next week...
  - Problem set #1 due!

Under the hood: 'optim()'

- How exactly does R fit this line?
- We know how linear regression works: we just square residuals and find line that minimizes this number
- Turns out there was actually a formula that found this solution without all the faff of experimenting:

$$\beta = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2}$$

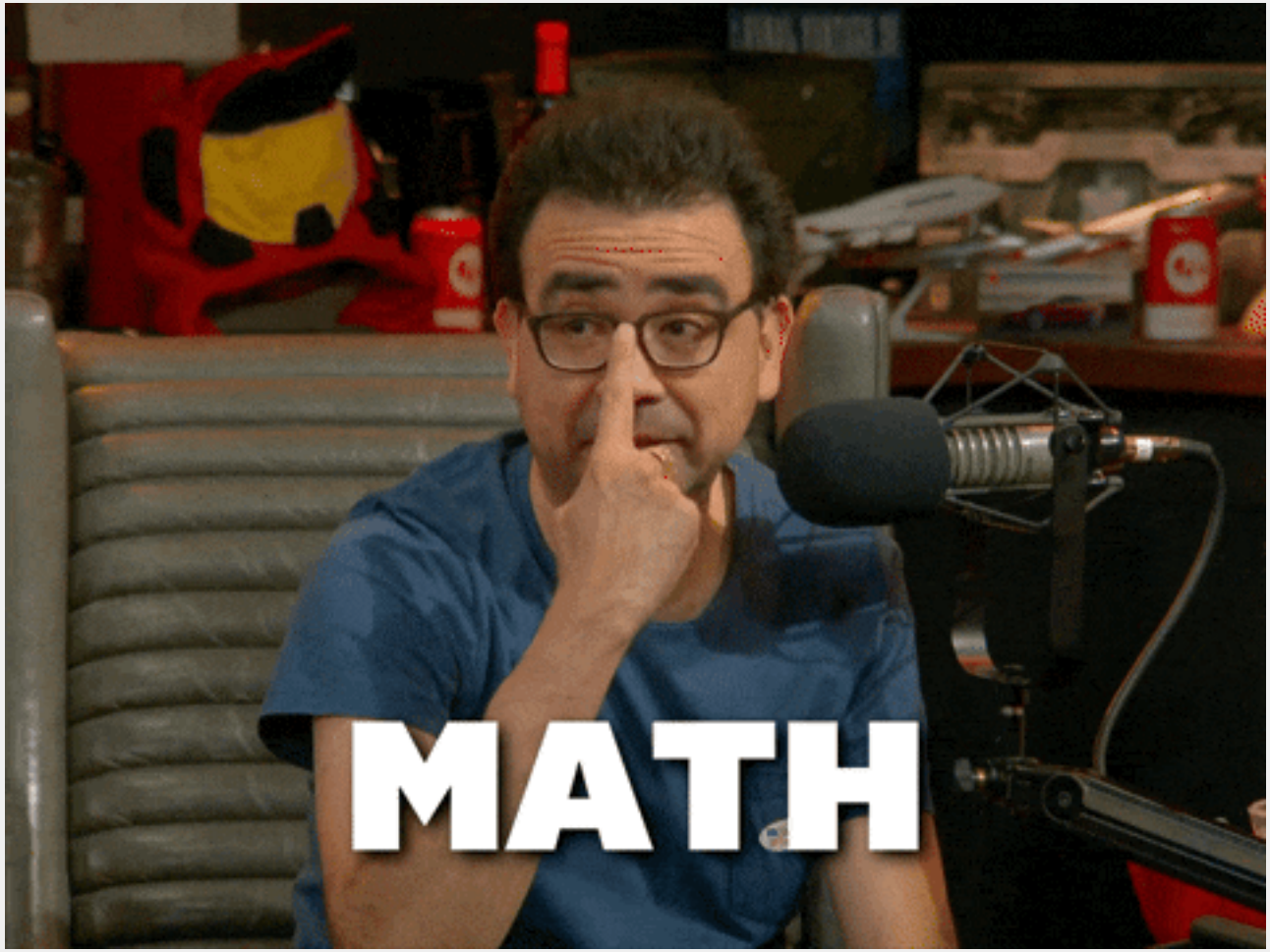
$$\alpha = \bar{y} - b\bar{x}$$

Under the hood: 'optim()'

- Wouldn't it be great if there was a similar equation for logistic regression?
  - Sadly, there is not...
- Let's watch a quick video that explains why

Under the hood: 'optim()'

- So, if there's no equation, how do we work out the best fitting line?



MLE setup - Define data: Predictors

- Let's see how it works in R
- We'll begin by setting a seed for our random data generating functions
- And, create a vector of predictor variables using `rnorm()`
- Then, we'll set the coefficients we want to estimate

```
# set seed so we all get same answers
set.seed(1234)
# create design matrix
# only 2 predictors, remember 1st column is 1s
X <- rnorm(100)
# define "real"/true relationship
real_beta <- 3
```

MLE setup - Define data: Outcome

- We want an outcome variable, *y*, which is binomial (1 or 0)
  - We're going to make the probability of a 1 or 0 dependent on a combination of the coefficients we just made up

```
# create output variables
# as linear function of covariates
# (1) binom
y_binom <- rbinom(100, 1, exp(X*real_beta)/(1+exp(X*real_beta)))
```

MLE setup - Likelihood function

Because we're using the binomial distribution, instead of the parameters being mean and standard deviation, we have probability (check `?rbinom( )` for details)

- This is the 3rd argument, which is what the MLE will be trying to maximise (estimate)
- Note the equation, which is the reverse of our link function, just like we saw before

```r
# derive our log-likelihood function for binomial distribution
binom_likelihood <- function(outcome, input, parameter) {
  # calculate probability of success on each trial
  p <- exp(parameter[1] + parameter[2]*input)/(1+exp(parameter[1] + parameter[2]*input))
  # access probability density function (pdf) for binomial distribution
  # specifically, calculate log.likelihood function
  # using sum and negative since its' log, not normal likelihood function
  -sum(dbinom(outcome, 1, p, log=TRUE))
}
```

MLE - optim() in R

Use the ‘optim()‘ function

- ‘optim()‘ minimises (or maximises) a function
- Here, we’ll supply the function we used to generate y_binom, but with the coefficients missing

```
# optimise our log-likelihood function
# need to put in par, which are initial values for parameters to be optimized over
# we'll start with zero and 1 for intercept and beta
# using BFGS because it's a quasi-Newton method
# so similar to what we did in class, and what you'll get from glm()
results_binom <- optim(fn=binom_likelihood, outcome=y_binom, input=X, par=0:1, hessian=T, method="BFGS")
# print our estimated coefficients (intercept and beta_1)
results_binom$par
```

```
[1] 0.1443736 3.6073485
```

MLE - optim() compared to glm()

Let's compare the results with the 'glm()' function

```r
# confirm that we get the same thing in with glm()
coef(glm(y_binom~X, family=binomial))
```

```
(Intercept)           X
  0.1443735   3.6073484
```

Overview

- Just learned: How to create our own glm() with MLE
- Now, let's practice deriving our own likelihood functions & executing models using MLE!
- Next week, we'll practice logistic regression on some real data