

# **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:

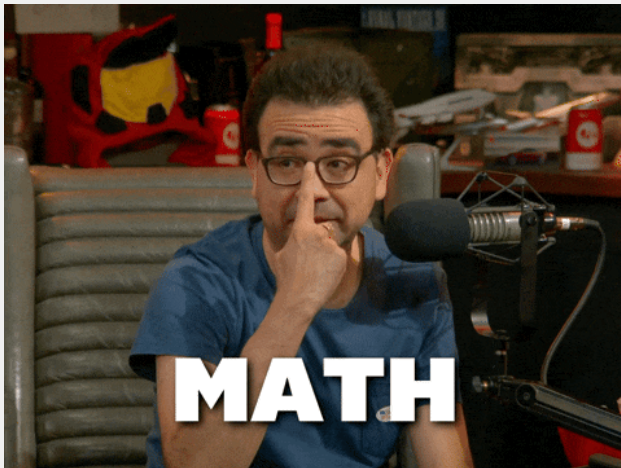
$$\beta = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2}$$
$$\alpha = \bar{y} - b\bar{x}$$

# 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
  - converting y axis into log value (negative to positive infinity)
  - taking log transformation of likelihood -> just adding probabilities together
  - rotating line -> changing the slope (trying to understand which fit is the best fit)
    - log likelihood is the same as likelihood score

## 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

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

# 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

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

using beta score and input value that we have, we want 100 observations to get a binomial distribution of  $y$

# 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

```
1 # derive our log-likelihood function for binomial distribution
2 binom_likelihood <- function(outcome, input, parameter) {
3   # calculate probability of success on each trial
4   p <- exp(parameter[1] + parameter[2]*input)/(1+exp(parameter[1] + parameter[2]*input))
5   # access probability density function (pdf) for binomial distribution
6   # specifically, calculate log.likelihood function
7   # using sum and negative since its' log, not normal likelihood function
8   -sum(dbinom(outcome, 1, p, log=TRUE))
9 }
```



# 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

irrespective of where you start, you arrive at a point where this is the best set of beta values

```
1 # optimise our log-likelihood function
2 # need to put in par, which are initial values for parameters to
  be optimized over
3 # we'll start with zero and 1 for intercept and beta
4 # using BFGS because it's a quasi-Newton method
5 # so similar to what we did in class, and what you'll get from glm
  ()
```

```
6 results_binom <- optim(fn=binom_likelihood, outcome=y_binom, input
  =X, par=0:1, hessian=T, method="BFGS")
```

```
7 # print our estimated coefficients (intercept and beta_1)
```

```
8 results_binom$par
```

tells us which set of parameters give us the max likelihood score for our function

- finds the minimum, not max, so we take the negative of it

```
[1] 0.1443736 3.6073485
```

if we don't take negative, we are looking for the minimum value

## MLE - OPTIM() COMPARED TO GLM()

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

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

(Intercept)	X
0.1443735	3.6073484

# OVERVIEW

likelihood function is very unimodal, it just has one maximum, so you could really take any number

- 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