Week 2
Tutorial - GLMs

Applied Statistical Analysis II

Lecturer: Jeffrey Ziegler, PhD
Teaching Fellow: Shekhar Kedia

ASDS - Trinity College Dublin

Spring 2026

Roadmap for today

- Today:
  - GLMs
    1. Know how to use the ‘glm()‘ function in R
    2. Understand a little about how GLM works, & how it differs from linear regression
    3. Regret not having learned more calculus ☺
- By next week, please...
  - Start problem set #1!

GLMs in R

- Today, look at **generalized linear model**
- GLM allows us to extend principle of linear regression to different kinds of outcomes using a *link function*
  - Simply an equation that "links" linear predictors to non-linear outcomes
  - Ex: "yes/no", counts, etc.

GLMs in R

Today's goal: More theoretical than usual for our tutorials

- Introduce the 'glm()' function in R
- Begin to see how it works 'under the hood' using what is known as **maximum likelihood estimation**
  - We'll focus on this explicitly next week
- MLE is just a way of estimating parameters of a distribution from the available data
  - For instance, in certain circumstances, OLS method we used last term satisfies objective of MLE

Comparative to what the OLS does - gets us our best parameters (unbiased, efficient)

- Helps us establish relationship between parameter and outcome

The ‘glm()‘ function

- ‘glm()‘ function is very similar to ‘lm()‘ function we’re already familiar with
- In fact, you can run a linear regression with ‘glm()‘ function, because linear regression is part of the same family as the generalised linear model

```r
lm(Sepal.Length ~ Sepal.Width + Species, data = iris)
```

```r
glm(Sepal.Length ~ Sepal.Width + Species, data = iris,
  family = "gaussian")
```

Still using normal distribution, hence Gaussian

THE ‘GLM()‘ FUNCTION

AIC & BIC = More to do with model complexity -> similar to R^2 and adjusted R^2; adding more predictors will likely show more variance being explained but its not necessarily a better model

	lm()	glm()
(Intercept)	2.25^{***}	2.25^{***}
	(0.37)	(0.37)
Sepal.Width	0.80^{***}	0.80^{***}
	(0.11)	(0.11)
Speciesversicolor	1.46^{***}	1.46^{***}
	(0.11)	(0.11)
Speciesvirginica	1.95^{***}	1.95^{***}
	(0.10)	(0.10)
R^2	0.73	
Adj. R^2	0.72	
Num. obs.	150	150
AIC		183.94
BIC		198.99
Log Likelihood		−86.97
Deviance		28.00

If these are all 0, you have a perfect model

- See how ‘glm()‘ function, when we specify the ‘family = ‘ argument to "gaussian", gives us *almost* same output as ‘lm()‘ function?
- This is because linear regression is based on normal, or Gaussian, distribution
- What do you notice is different though?

Check validity of model -> run different covariates

Log likelihood = How much variation is explained by the model
deviance = how much model deviates from being perfect

THE ‘GLM()‘ FUNCTION

- For example, what if we want to use ‘glm()‘ to perform regression for binary outcome?
- We simply specify distribution for which logistic regression estimates parameters, i.e., binomial distribution

```r
glm(am ~ cyl + disp, data = mtcars, family = "binomial")
```

Logistic (or logit) regression

Quickly review what we use logistic regression for, and how it works

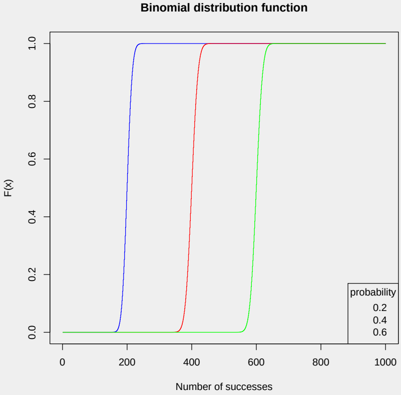
- We have an outcome that can only be true or false, yes or no
- Then, our estimate for this outcome needs to be bounded between zero (for false/no) and one (for true/yes)
- It would make no sense to have an estimate less than zero, or more than 1
- Moreover, if we did get such an estimate, it would suggest that our model was somehow wrong, and that all our other predictions might be off

Logistic (or logit) regression

Basically, we need a distribution that falls between zero and one

- And, we need to be able to link coefficients of our predictors to this distribution

Here’s the distribution:



Here’s the link function:

Odds ratio

$$\log \frac{p}{1-p} = \beta X$$

Taking a log of the odds -> related to beta values in a linear form

Example GLM: Logit regression

- Remember: All link function does is convert from a probability (0 to 1 scale), which has a curve shape, to log odds ($-\infty$ to ∞ scale), which is a straight line
- Typically use log odds for βX (literally, the log of the odds) because these are symmetrical and normally distributed
- Taking log of the odds also converts terms in our model from a multiplicative to an additive relationship (i.e., we add terms rather than multiply them)
  - Which is similar to what we are familiar with from linear regression

Example GLM: Logit regression

- Easy to get confused in logistic regression
- There is a difference between odds and odds ratio, log (odds) and log (odds ratio)
  - Logit link function used to convert between probabilities and log odds
  - Log-likelihood transform used in maximum likelihood estimation to fit predicted line
- Takes time to familiarize yourself with these
- In general, we'll focus on how to interpret our model coefficients and how to make a prediction based on them

Using logistic regression

Let’s compare a logistic regression model with a linear model we’re familiar with, using the ‘iris‘ dataset we’re also familiar with

```r
dat <- iris
dat$set <- ifelse(iris$Species == "setosa", 1, 0)
mod1 <- lm(set ~ Petal.Length + Petal.Width, data = dat))
```

Binary outcome

	Dependent variable:
	set
Petal.Length	-0.251^{***}
	(0.032)
Petal.Width	0.010
	(0.073)
Constant	1.266^{***}
	(0.044)
Observations	150
R^2	0.852
Adjusted R^2	0.849
Residual Std. Error	0.183 (df = 147)
F Statistic	421.497^{***} (df = 2; 147)

Using logistic regression

- So, we ran a linear model on a binary outcome (is iris of species setosa or not?)
- Let's make a prediction from our model based upon some imaginary data
  - An iris with petal length of 5.4cm and width of 2.4cm

```r
newdat <- data.frame(Petal.Length = 5.4, Petal.Width = 2.4)
predict(mod1, newdat)
```

```
1
-0.0675413
```

Should be 0 or 1 but lm() is not bounded by 0 or 1

Using logistic regression

As we can see, our model gives us a prediction below zero - does this make sense?

Let’s try again with logistic regression:

```r
mod2 <- glm(set ~ Petal.Length + Petal.Width, data = dat, family =
      "binomial")
```

	Dependent variable:
	set
Petal.Length	−17.596
	(43,449.070)
Petal.Width	−33.887
	(115,850.800)
Constant	69.447
	(43,042.940)
Observations	150
Log Likelihood	−0.000
Akaike Inf. Crit.	6.000

Using logistic regression

- So, we get a different set of coefficients for our predictor variables, both of which are now negative...
  - What does this mean?
  - How do we make a prediction?
- Let's try with our same imaginary data

```r
predict(mod2, newdat)
```

```
1
-106.8992
```

Using logistic regression

Well, that doesn't seem too useful - we're even further below zero than the linear model

Except, if we check the help file for ‘predict.glm()‘...

Let's try again with the argument `type = "response"`

```r
predict(mod2, newdat, type = "response")
```

This is the p value

```
1
2.220446e-16
```

Using logistic regression

- What's the difference?
- By default we get log odds prediction
  - If we want to get the probability from this, we need to reverse the link function, which is...

$$p = \frac{e^{\log(odds)}}{1 + e^{\log(odds)}}$$

Using logistic regression

Let's try this out on a slightly more normal range of data, petal length = 2.3 and petal width = 0.8

```r
newdat2 <- data.frame(Petal.Length = 2.3, Petal.Width = 0.8)
predict(mod2, newdat2, type = "response")
```

```
1
0.8661015
```

```r
log_odds <- predict(mod2, newdat2)
exp(log_odds)/(1+exp(log_odds))
```

```
1
0.8661015
```

```r
1/(1+exp(-log_odds))
```

```
1
0.8661015
```

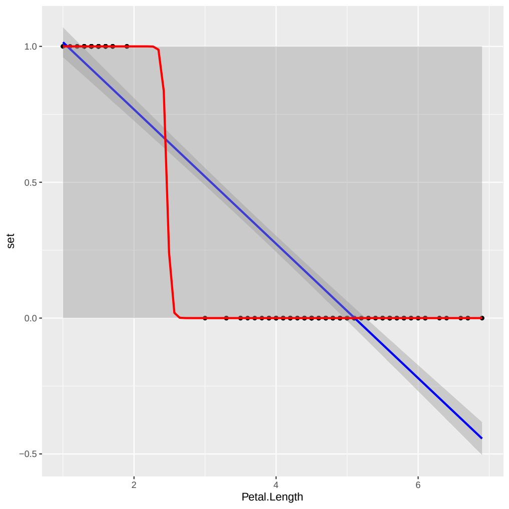
Using logistic regression

- As you can see, we get same prediction!
- Interpreting coefficients and predicted values in logit regression is less straightforward than linear regression
  - We'll look into this in greater detail in the weeks to come
- This site has some helpful examples that explain difference between log odds, odds, and probability, and how we can calculate each with R

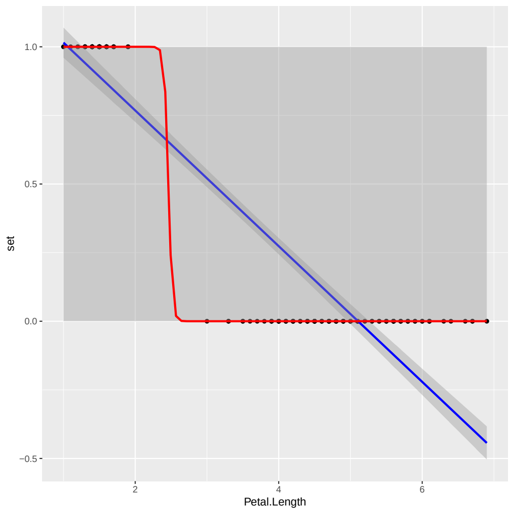
Linear vs Logit regression

Let’s plot probability of an iris being from setosa species according to length of its petals

```
ggplot(data = dat, aes(
    Petal.Length, set)) +
geom_point() +
geom_smooth(method = "lm",
     color = "blue") +
geom_smooth(method = "glm"
    , method.args = list(
    family = "binomial"),
    color = "red")
```



Linear vs Logit regression



Larger petal -> less likely to be setosa iris

- As we can see, the larger the petal, the lower the probability
- However, linear regression line is not bounded by zero on the y axis
  - So, line moves negative above roughly 5cm on x-axis
- Whereas, logit regression is asymptotic to zero and one on y-axis
  - No matter how large the petal, prediction does not go below

Overview

- Right, that's enough calculus for today
- Now, let's practice organizing data, executing a lm() + glm(), and comparing output
- Next week, we'll figure out how to create our own glm with MLE
- Then in week 4, we'll practice logistic regression on some real data