Class 3: Linear Regression and GLMs

Andrew Parnell
andrew.parnell@ucd.ie

Maynooth University
National University of Ireland Maynooth

Learning outcomes

- Know how to fit and interpret a linear regression
- Know the difference between a linear regression model and a generalised linear model (GLM)
- Know what a link function is and why it is used
- Be able to interpret the output of a simple GLM

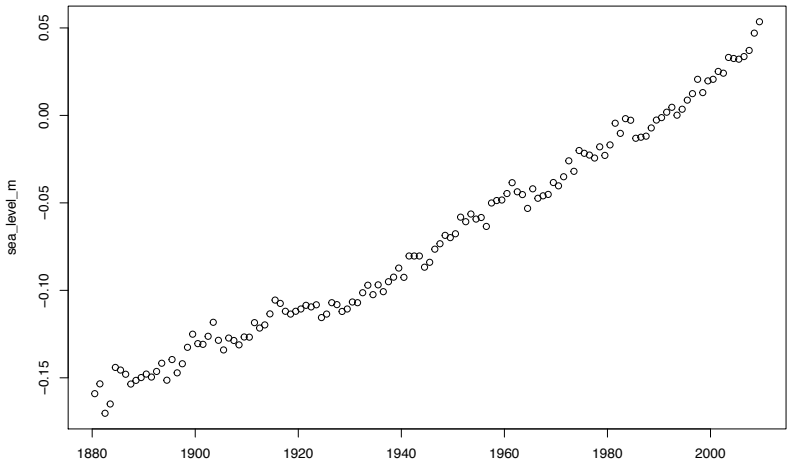
Using more of the data

- It's very rare that we are given a data set with just a single variable
- More often we're given multiple variables and asked to predict one or more of the variables from the others
- This is an example of *conditional* inference
- It might look more complicated, but this is still just fitting a probability distribution to some data

Linear regression example

Here is some data on sea level rise:

```
sl = read.csv('../data/tide_gauge.csv')
with(sl, plot(year_AD, sea_level_m))
```



Linear regression models

- The simplest version of a linear regression model has:
  - A *response variable* (y) which is what we are trying to predict/understand
  - An *explanatory variable* or *covariate* (x) which is what we are trying to predict the response variable from
  - Some *residual uncertainty* (ϵ) which is the leftover uncertainty that is not accounted for by the explanatory variable
- Our goal is to predict the response variable from the explanatory variable, *or* to try and discover if the explanatory variable *causes* some kind of change in the response

The linear models in maths

- We write the linear model as:

$$y_i = \alpha + \beta x_i + \epsilon_i$$

  where α is the intercept, β the slope, and $i = 1, \ldots, N$ represents each of the N observations
- Usually we make the additional assumption that $\epsilon_i \sim N(0, \sigma^2)$ where σ is the residual standard deviation
- Writing this in probability distributions:

$$y_i | x_i, \alpha, \beta, \sigma \sim N(\alpha + \beta x_i, \sigma^2)$$

.

Fiting linear regression models

- We can create a likelihood as before by guessing some values of the parameters and then using the `dnorm` function to compute the likelihood value

```
alpha = 2
beta = 1.5
sigma = 0.6
y = sl$sea_level_m
x = sl$year_AD
sum(dnorm(y, mean = alpha + beta*x, sd = sigma,
          log = TRUE))
```

```
## [1] -1539607458
```

- Not a very high value of the likelihood!

Finding the best values

- Luckily, R has the `lm` function to find the best fitting values of the parameters

```
summary(lm(y ~ x))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##       Min         1Q     Median
## -0.0167787 -0.0051874 -0.0003646
##         3Q        Max
##  0.0063022  0.0252621
##
## Coefficients:
##               Estimate Std. Error
## (Intercept) -3.062e+00  3.937e-02
## x            1.538e-03  2.024e-05
##             t value Pr(>|t|)
## (Intercept)  -77.78   <2e-16 ***
## x             75.99   <2e-16 ***
## ---
## Signif. codes:
##   0 '***' 0.001 '**' 0.01 '*' 0.05
##   '.' 0.1 ' ' 1
##
## Residual standard error: 0.008659 on 128 degrees of freedom
## Multiple R-squared:  0.9783, Adjusted R-squared:  0.9781
## F-statistic:  5775 on 1 and 128 DF,  p-value: < 2.2e-16
```

Checking the likelihood

```
alpha = coefficients(lm(y ~ x))[1]
beta = coefficients(lm(y ~ x))[2]
sigma = summary(lm(y ~ x))$sigma
sum(dnorm(y, mean = alpha + beta*x, sd = sigma,
          log = TRUE))
```

```
## [1] 433.9335
```

A much higher value of the likelihood!

Other notes about `lm`

- Usually we would store the output from `lm` in another object to allow us to manipulate the output, e.g. `my_model = lm(y ~ x)`
- We can use the `confint` function to get confidence intervals on the parameters
- We can predict future values of sea level from the model by giving it new x values, e.g.

```
my_model = lm(y ~ x)
predict(my_model, newdata = data.frame(x = 2050))
```

```
##          1
## 0.09055375
```

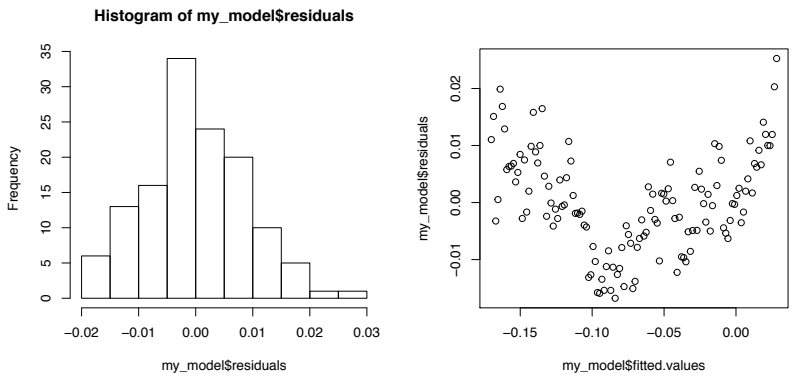

Checking the model

- Just finding the best values of the parameters and their uncertainty is not the whole story
- We need to check the fit of the model
- We can do this by analysing whether the assumed probability distribution is correct or not
- Can look at the probability distribution we have fitted to the data, but most people just look at the leftover bits - the residuals

Residual plot

- Common to plot a histogram of the residuals, and a scatter plot of the residuals vs the fits

```
par(mfrow=c(1, 2))
hist(my_model$residuals)
plot(my_model$fitted.values, my_model$residuals)
```



Transforming the data

- Sometimes the residuals of a linear regression look a little bit mis-shapen
- We might improve the fit by adding more covariates, or by transforming the data (the response and/or the covariates)
- If your variables have very large values then you might get better results by standardising your data (subtracting the mean and dividing by the standard deviation)
- Common transformations include the log or square root
- A common transformation in time series data is the *Box-Cox* transformation...

Box-Cox

- The Box-Cox transformation is:

$$f(x;\lambda) = \frac{x^\lambda - 1}{\lambda} \text{ if } \lambda \neq 0$$

or

$$f(x;\lambda) = \log(x) \text{ if } \lambda = 0$$

- The usual reason to use it is when the data are *skewed* and we want it to look more symmetrical
- You need to choose the value of λ; usually trial and error

From LMs to GLMs

- If a normal distribution is not suitable for the residuals we need to choose another probability distribution
- Here is some data from an experiment on whitefly:

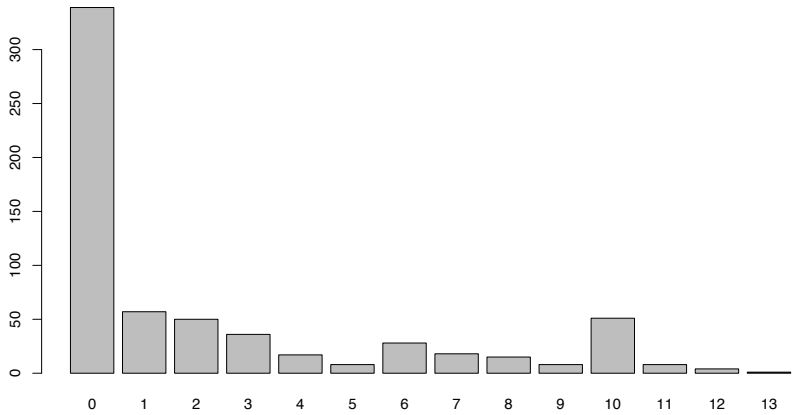
```
whitefly = read.csv('../data/whitefly.csv')
head(whitefly, 4)
```

```
##   imm week block trt  n live plantid
## 1  15    1     3   5 12   11       1
## 2  16    2     3   5  8    6       1
## 3  28    3     3   5 10   10       1
## 4  17    4     3   5 10    8       1
```

- The `live` and `n` columns indicate how many whitefly survived and were used in the experiment respectively
- We have a fixed total and a number of surviving whitefly out of this total. Which probability distribution might be appropriate?

Plotting the whitefly data

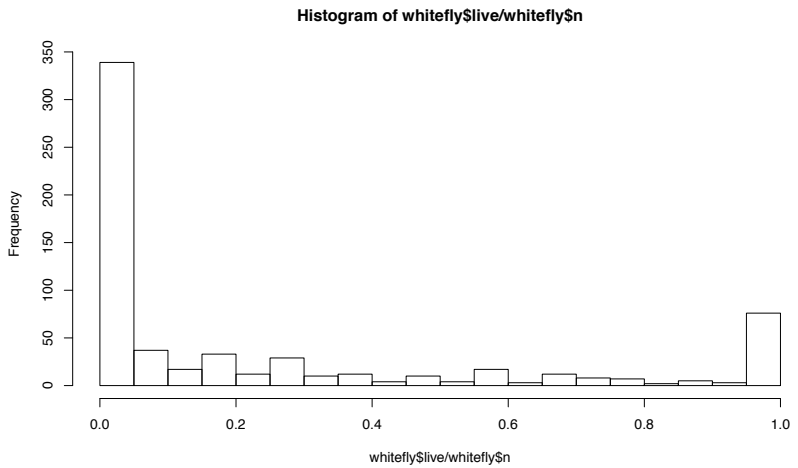
```
barplot(table(whitefly$live))
```



- Doesn't really show the whole story as we're ignoring the total

Second go

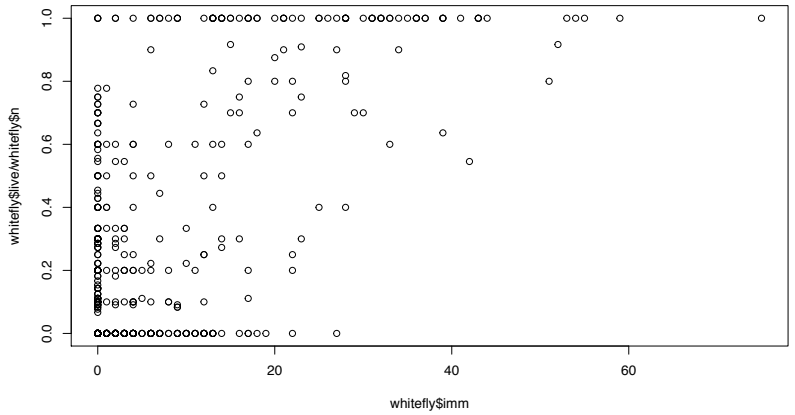
```
hist(whitefly$live/whitefly$n, breaks = 30)
```



- Better, but actually there is also a covariate in the number of immature whitefly that were included (variable `imm`)

Third go

```
plot(whitefly$imm, whitefly$live/whitefly$n)
```



- Looks like the more immature whitefly there were, the more likely they were to survive

Fitting a model

- Let's call y the number of live whitefly at the end of the experiment, and n the number of whitefly used in the experiment
- Let's fit a binomial distribution:

$$y \sim Bin(n, p)$$

- We know the value of n so it is not really a parameter but a fixed part of the data
- We need to estimate p
- We could use method of moments or maximum likelihood. If we use method of moments we get $\hat{p} = 0.245$

Fitting a better model

- What if we wanted to include `imm` as a covariate?
- One way would be to let $p = \alpha + \beta x$ where x is the number of immature whitefly
- We could fit this using maximum likelihood to get estimates of $\hat{\alpha}$ and $\hat{\beta}$
- This is now a *Generalised Linear Model* (GLM)
- The likelihood would be:

```
y = whitefly$live
n = whitefly$n
x = whitefly$imm
alpha = 0.5
beta = 0.0001
sum(dbinom(y, n, alpha + beta * x, log = TRUE))
```

```
## [1] -3330.495
```

Fitting an even better model

- Suppose after using maximum likelihood we got $\hat{\alpha} = 0.2$ and $\hat{\beta} = 0.01$. Then the likelihood would be

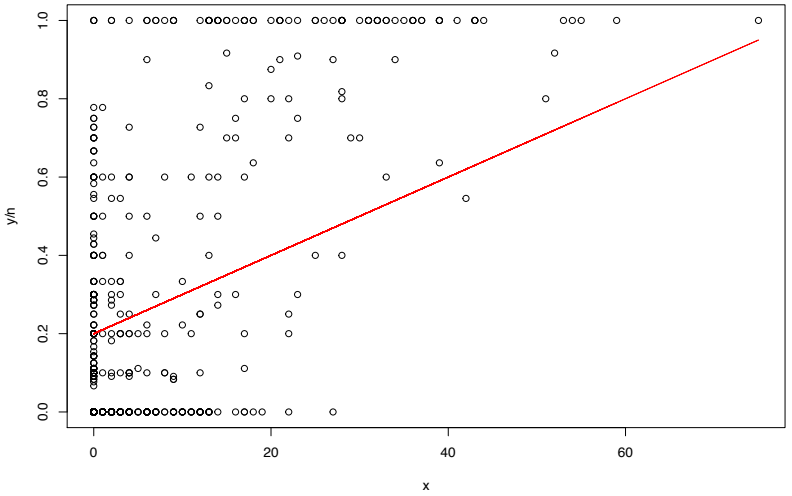
```
alpha = 0.2
beta = 0.01
sum(dbinom(y, n, alpha + beta * x, log = TRUE))
```

```
## [1] -1971.889
```

- Can anyone see any problems with this model? (Hint: suppose I wanted to predict what proportion would die when $x = 100$)

Plotting the fitted model

```
plot(x, y/n)
lines(x, alpha + beta * x, col = 'red')
```



The logit function

- To stop the lines going out of the range (0, 1) people often use the *logit* transformation:

$$\log\left[\frac{p}{1-p}\right] = \alpha + \beta x \text{ or } p = \frac{e^{\alpha+\beta x}}{e^{\alpha+\beta x}+1}$$

- The latter is known as the inverse logit function

- We now maximise the likelihood:

```
p = exp(alpha + beta * x) / (exp(alpha + beta * x) + 1)
sum(dbinom(y, n, p, log = TRUE))
```

```
## [1] -3630.322
```

- These logit and inverse logit functions are in the boot package:

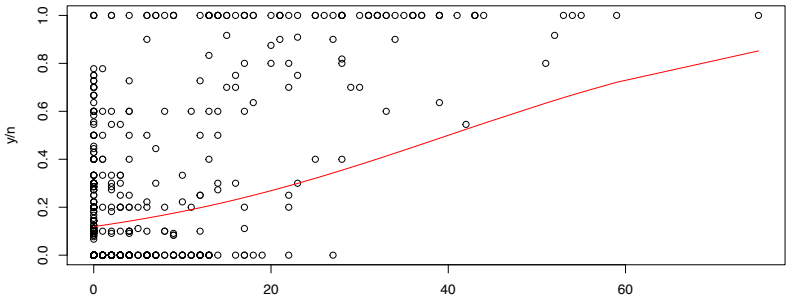
```
library(boot)
sum(dbinom(y, n, inv.logit(alpha + beta * x), log = TRUE))
```

```
## [1] -3630.322
```


Plotting the fit

- Suppose under this method we got maximum likelihood estimates of $\hat{\alpha} = -2$ and $\hat{\beta} = 0.05$
- A plot of the fitted values is now:

```
alpha = -2
beta = 0.05
o = order(x)
plot(x, y/n)
lines(x[o], inv.logit(alpha + beta * x[o]), col = 'red')
```



Finding the maximum likelihood values

- R has a function called `glm` to find the maximum likelihood values for us
- For binomial model with a logit *link function* we would type:

```
glm(cbind(y, n) ~ x, family = binomial(link = logit))
```

```
##
## Call:  glm(formula = cbind(y, n) ~ x, family = binomial(link
##
## Coefficients:
## (Intercept)            x
##    -1.95517      0.05391
##
## Degrees of Freedom: 639 Total (i.e. Null);  638 Residual
## Null Deviance:       2389
## Residual Deviance: 1688  AIC: 2541
```


Further details about the fit

- The interpretation of the $\hat{\beta}$ value (the coefficient of x) is in terms of *log odds*. A unit increase in x gives a $\exp(\hat{\beta}) \approx 1.06$ times increase in the probability of a whitefly surviving
- You'll see amongst the output something called the *deviance*. This is minus twice the log-likelihood
- It's a common measure used to compare models as the deviance for a linear regression model is just the mean square error
- Next to it you'll see the *Akaike Information Criterion* or AIC value, which penalises the deviance by adding on twice the number of parameters (i.e. a measure of the complexity of the model)
- Often, you would fit multiple models with different covariates and choose the one(s) with the smallest AIC

Other glms

- Another common glm is the Poisson, useful for count data
- For example, suppose we treated the number of whitefly who survived as a count, and temporarily ignored the n values
- We could fit:

```
glm(y ~ x, family = poisson(link = log))
```

```
##
## Call:  glm(formula = y ~ x, family = poisson(link = log))
##
## Coefficients:
## (Intercept)           x
##     0.38559      0.04608
##
## Degrees of Freedom: 639 Total (i.e. Null);  638 Residual
## Null Deviance:       3055
## Residual Deviance: 2154  AIC: 3124
```

- Recall that the parameter in the Poisson probability distribution represents the mean (and the variance) which must be positive.
- Like the logit, the log link stops the rate parameter from going negative

A final word on glms

- There are lots of different types of GLMs We can do Gamma, Negative Binomial, Beta, Inverse Gaussian, ...
- Each has a link function which transforms the main parameter into an unrestricted range through which we can include covariates
- It's also simple to include extra covariates or interactions:

```
glm(y ~ x1 + x2 + x1:x2, family = poisson(link = log))
```

- Residual checks are still important, and R will create them for us
- We can get at them via e.g.

```
my_model = glm(cbind(y, n) ~ x,
               family = binomial(link = logit))
plot(my_model)
```

Summary

- Linear Regression and Generalised Linear Models are two common ways to extend standard probability distributions to include covariates
- We estimate the parameters via maximum likelihood using e.g. `lm` or `glm`
- We sometimes need to include a link function which transforms the parameters into an unrestricted range
- There are lots of different types of GLM for every flavour of probability distribution