

# Logistic regression - introduction

Ian Handel

2019-06-25

## Introduction

THESE NOTES are a basic introduction to binary logistic regression used to analyse data with binary outcomes. In these notes we'll aim to cover the following learning outcomes:

- **Refresher** on logs, odds, probability and linear regression
- Understand why linear regression not sensible for **binary data**
- Explain how **logit** and binomial model let us **extend linear regression**
- Be able to run a **simple logistic regression in R**
- Be able to explain basic R glm **output**
- Be able to explain **estimates** with categorical and continuous variables
- Explain **significance test results** on variables
- Introduce some basic ideas for **selecting variables and models**
- **Things to watch out for!**
- **Know where to go next!**

## Prerequisites

We'll cover a couple of background topics but to follow these notes you'll need to be able to load packages in R, run simple code in R and have a basic understanding of linear regression and statistical hypothesis tests.

## Running the example code

To run the R code in these notes you'll need to start an rstudio project, load in the example data-set and have a few packages downloaded and loaded into your R session.

To download the packages (if you don't have them already) use...

```
install.packages("tidyverse")
install.packages("boot")
install.packages("broom")
install.packages("skimr")
install.packages("sjPlot")
install.packages("kableExtra")
```

You can also use the install button in the RStudio packages tab to install these

Start a new project in rstudio and in an rscript use the following code to load the libraries you need and to import/load the data...

Loading the packages...

```
library(tidyverse)
library(boot)
library(broom)
library(skimr)
library(sjPlot)
library(kableExtra)
```

Loading the data (from the csv file on Basecamp)...

```
dat <- read_csv("logreg_data_01_20190530.csv")
```

Once loaded you should see the a 'dat' object in the RStudio environment tab (usually in the top-right of your screen)

This dataset describes 500 animals giving their weight in kg, age in days, supplement levels (mg), sex, region where they live (A, B, C or D) and whether they were treated with anthelmintics or not. It's a dataset made up for this course by the way!

## Revision / background topics

### Logarithms ('logs')

Skip this if you are happy with logs (including base 'e')

'Logs' are a mathematical function that changes a number. They take the form  $\log_b(x) = y$ . What this means is  $b$  to the power  $y$  will give us  $x$ . It's easier to understand with examples. Let's start with base 10...

$$\log_{10}(10) = 1$$

$$\log_{10}(1000) = 3$$

$$\log_{10}(0.01) = -2$$

So the logs of all the numbers in brackets are the number you'd need to raise 10 to to get them.

We can have other bases e.g.  $e$ ,  $e$  is a special mathematical constant that features a lot behind the scene in statistics it's roughly 2.718...

$$\log_e(2.718) \simeq 1$$

'Inverse logs' let us turn logs back into the original number. We simply raise the 'base' of our logs to the number we want to invert and we end up with the original number. So  $\log_{10}(1000) = 3$  and  $10^3 = 1000$ ...

One feature of logs is that adding the logs of two numbers is equivalent to multiplying the numbers...

$$100 \times 1000 = 100000$$

$$\log_{10}(100) + \log_{10}(1000) = \log_{10}(100000)$$

Because  $\log_{10}(100) = 2$ ,  $\log_{10}(1000) = 3$  and  $\log_{10}(100000) = 5$

If this all seems a bit too much don't worry. Just remember that adding logs is like multiplying numbers and you'll be fine!

## Odds and probability

**Probabilities** have values from 0 ('never happens') to 1 ('always happens')

**'events of interest' ÷ 'all events'.**

What is the probability that a fair coin lands on heads?

$$1/2 = 0.5$$

What is the probability that a 6 sided die lands on 4?

$$1/6 \simeq 0.166$$

**Odds** have values from 0 ('never happen') to infinity ('always happens')

**'events of interest' ÷ 'other events'.**

What are the odds that a fair coin lands on heads?

$$1/1 = 1$$

What are the odds that a 6 sided die lands on 4?

$$1/5 = 0.2$$

## Linear regression

Remember that linear regression is a statistical method that lets us understand and predict numerical outcomes using one or more predictor variables. The predictor variables may be numerical or categorical. Linear regression also assumes there's a linear i.e. straight line relationship between the predictor numerical variable and the outcome. Using the data set we have loaded we can plot weight against age and sex and see that there looks to be a linear relationship between age and weight for each sex...

```
ggplot(dat) +
  aes(age, weight, colour = sex) +
  geom_point(shape = 1) +
  geom_smooth(method = "lm", se = FALSE) +
  theme_bw() +
  labs(x = "Age (days)",
       y = "Weight (Kg)",
       colour = "Sex")
```

In R we can use the `lm()` function to fit a linear model to this data. Normally we store the results of using the function in an R object and look at it using `'summary()'`. We can also get a tidier output using `get_model_data()` from the `sjPlot` package. Here we make a linear model predicting the animal's weight from their age (a numerical variable) and their sex (a categorical variable).

```
mod1 <- lm(weight ~ age + sex, data = dat)
```

Using `summary()` from base-R...

```
summary(mod1)
```

Call:

```
lm(formula = weight ~ age + sex, data = dat)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-2.3506	-0.4732	-0.0509	0.4572	2.0223

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	49.93305	0.40486	123.33	<2e-16 ***
age	0.12045	0.00191	63.07	<2e-16 ***
sexmale	3.51683	0.06541	53.76	<2e-16 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

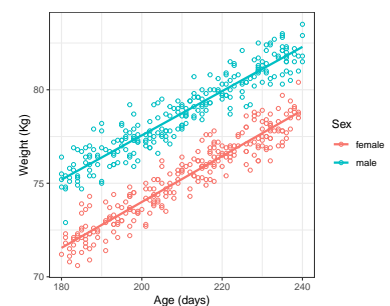


Figure 1: Weight vs Age of the animals

Residual standard error: 0.7312 on 497 degrees of freedom  
 Multiple R-squared: 0.932, Adjusted R-squared: 0.9317  
 F-statistic: 3404 on 2 and 497 DF, p-value: < 2.2e-16

Using `get_model_data()` from `sjPlot` package and selecting the output columns we want...

```
get_model_data(mod1) %>%
  select(term:p.stars) %>%
  print()
```

```
# A tibble: 2 x 8
  term      estimate std.error statistic   p.value conf.low conf.high p.stars
<fct>      <dbl>     <dbl>     <dbl>   <dbl>   <dbl>   <dbl> <chr>
1 age         0.120    0.00191     63.1 2.50e-239  0.117   0.124 ***
2 sexmale     3.52     0.0654     53.8 2.87e-209  3.39    3.65  ***
```

The analysis suggests that both **age** and **sex** are significant predictors of weight. Weight increasing by about 0.12 Kg per day of age and that males are about 3.5 Kg heavier than females. Right - now to look at binary (yes/no) outcome data and logistic regression...

## Analysing binary data

Binary data is common in epidemiological studies e.g. disease status (diseased or healthy), life (alive or dead) etc. In the example dataset we have the column `status` where animals can be either **healthy** or **diseased**.

ID	treatment	age	region	supp	sex	weight	status
A0049	control	221	D	7.891	male	80.9	diseased
A0485	control	220	A	2.651	male	79.5	diseased
A0321	treated	238	B	1.671	male	82.2	diseased
A0153	treated	183	C	6.346	female	71.7	healthy
A0074	treated	187	A	9.655	male	77.0	diseased
A0228	control	206	C	8.046	female	74.7	diseased

As we know that some animals were treated with an anthelmintic and some were not it may be interesting to look at the number of healthy and diseased animals in the treated and control (untreated) groups.

treatment	diseased	healthy
control	154	44
treated	209	93

Approximately 78% of the untreated (control) animals were diseased and 69% of the treated animals were diseased. We can use a 'Fisher's Exact test' to test if the proportion of diseased animals in the treatment vs control groups are significantly different. This test also estimates the odds ratio of disease between the two groups.

You can use `?fisher.test` in the RStudio console to get help on this function. Here we have also used the `with()` function to make it easier to use the 'dat' dataset and refer to the columns in the `fisher.test()` code

```
with(dat,
      {{fisher.test(status, treatment)}})
```

### Fisher's Exact Test for Count Data

```
data: status and treatment
p-value = 0.04032
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 1.010694 2.419528
sample estimates:
odds ratio
 1.556051
```

The Fisher's exact test reports a p-value, in this case 0.04. If this is less

than your chosen critical p-value you have evidence that the proportion of diseased animals is significantly different between the treated and untreated (control) animal groups. Fisher's Exact test is useful for comparisons between a binary outcome and a single, binary predictor. However we may want to look at the association between several predictors and a binary outcome in one go. This is called **multivariable** analysis and you'll have met it in linear regression. We can't look at this with a simple test so need to consider regression approaches...

## Multivariable analysis

We could convert our `status` column to a number coding healthy animals as **0** and diseased animals as **1** and then use conventional linear regression with the variables we want to consider as predictors. The following code will make a new column, `status01` recoding status to a number.

```
dat <- dat %>%
  mutate(status01 = as.numeric(status == "diseased"))
```

ID	treatment	age	region	supp	sex	weight	status	status01
A0001	treated	219	D	6.731	female	76.2	diseased	1
A0002	treated	218	C	7.950	female	76.0	diseased	1
A0003	control	214	A	5.617	female	75.3	healthy	0
A0004	control	194	A	8.490	female	72.1	diseased	1

Now we have converted the binary data into a number (0 or 1) we **could** try using linear regression. First, to help understand what we are trying here, we can plot the outcome against the treatment group and age of the animals we see this...

To understand how this works remember that `==` in R compares things and if they are the same returns TRUE and if they aren't returns FALSE. Then `as.numeric()` will convert FALSE to a 0 and TRUE to 1. It's a handy way of making binary text /factor data into 0's and 1's

```
ggplot(dat) +
  aes(age, status01, colour = treatment) +
  geom_point(shape = 1) +
  theme_bw(base_size = 16) +
  labs(x = "Age (days)",
       y = "Outcome",
       colour = "Treatment")
```

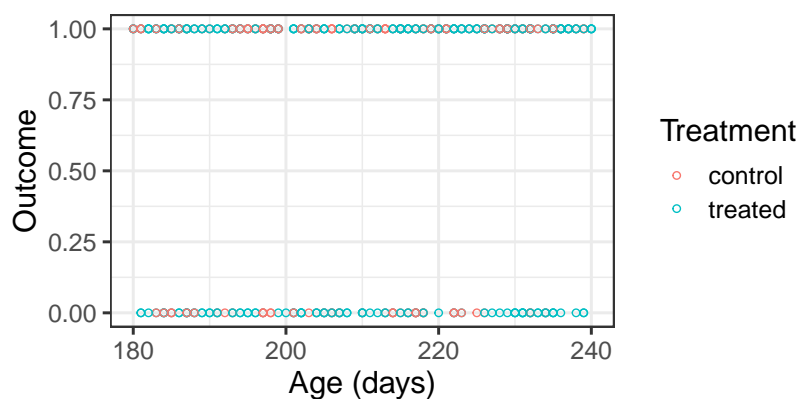


Figure 2: Status vs Age and treatment



Linear regression will attempt to fit straight lines of 'prediction' to the control and treatment data.

```
mod_lin <- lm(status01 ~ age + treatment, data = dat)
```

```
get_model_data(mod_lin) %>%  
  select(term:p.stars)
```

```
# A tibble: 2 x 8
```

	term	estimate	std.error	statistic	p.value	conf.low	conf.high	p.stars
	<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1	age	0.00203	0.00116	1.75	0.0811	-0.000246	0.00430	" "
2	treatmenttreated	-0.0876	0.0406	-2.16	0.0316	-0.167	-0.00796	*

Before we get too excited by this let's look at the fitted prediction line on the original plot.

```
dat %>%  
  modelr::add_predictions(mod_lin) %>%  
  ggplot() +  
  aes(age, status01, colour = treatment) +  
  geom_point(shape = 1) +  
  geom_line(aes(y = pred)) +  
  theme_bw(base_size = 16) +  
  labs(x = "Age (days)",  
       y = "Outcome",  
       colour = "Treatment")
```

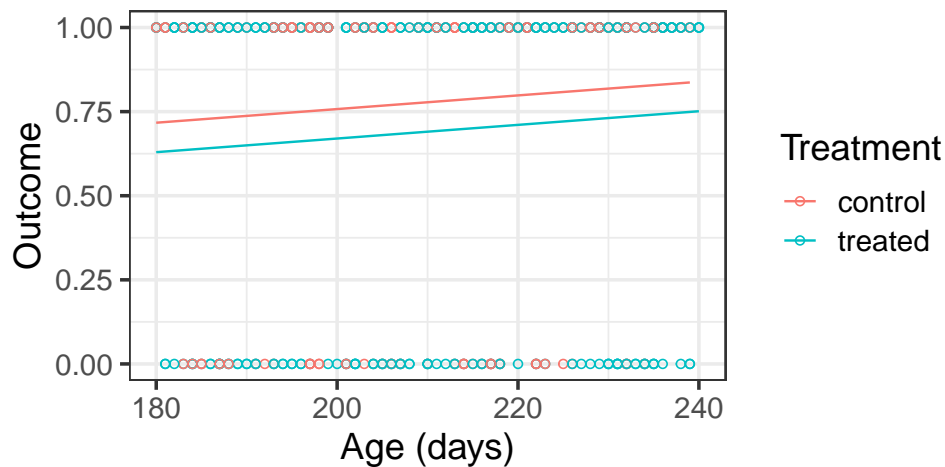
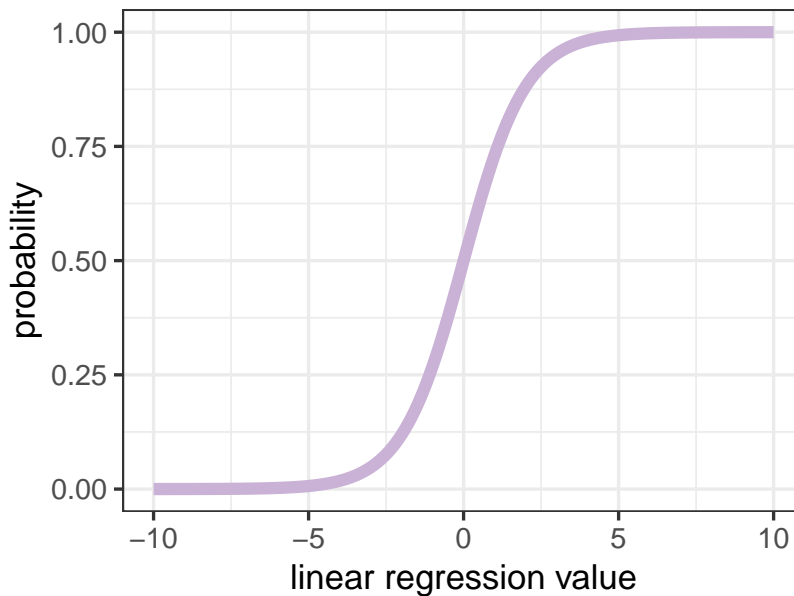


Figure 3: Status vs Age and treatment (with linear model)

Our outcome can only be **0** i.e. **healthy** or **1** i.e. **diseased**. However our linear model is predicting numbers in between. In some circumstances trying to use a linear model to predict binary 0/1/ data will even give values less than zero and more than one. We need to re-think the regression approach so that our model makes sense with an outcome that can only be **0** or **1** (or healthy / diseased etc).

We do this in logistic regression by making a model that estimates the **probability** of an animal having the outcome. Remembering that probabilities can take a value between **0** and **1** we modify the linear regression model so that its predictions lie only between 0 and 1. As linear regression can produce numbers ranging from hugely negative (negative infinity in fact) to hugely positive (positive infinity) we use a **logit** function.

Here's a plot of what it does...



It's turning a number (in this case from -10 to 10) into a probability that can **only** range from 0 to 1. If we expanded the range of numbers to be from minus infinity to plus infinity the probability would still only go from 0 to 1. The number is in fact the 'log' odds of the event. Logs of odds can range from - infinity to + infinity - because odds can range from 0 to infinity [Don't worry if you don't follow that maths!].

#The logistic regression models

So we use this logistic function together with a linear regression model to build a model that predicts the probability of an outcome given one or more predictor variables. The linear model bit generates a number from - infinity to + infinity and this is turned into a probability (0 - 1). So a model for disease based on age and treatment mathematically looks like this...

Note: In some circumstances using a linear model with a 0/1 outcome is a sensible strategy. It's called a 'linear probability model'. You'll see it used by economists. However we'd suggest you use logistic regression unless you are sure that a linear model with binary outcomes is correct!

Mathematically the regression value is  $\log_e\left(\frac{P}{1-P}\right)$  where  $P$  is the probability of the outcome. Understanding this isn't critical but if you want to read more about this look at the logistic regression chapter in the Dahoo book (see references)

$$\log_e\left(\frac{prob}{1-prob}\right) \sim \beta_0 + \beta_1 age + \beta_2 treatment$$

Or in words: The log of the odds of an animal being diseased are modelled by a linear combination of the predictor variables.

Let's look at a worked example using the data we have loaded...

## Worked example in R

First a reminder about the data structure...

```
head(dat)
```

ID	treatment	age	region	supp	sex	weight	status	status01
A0001	treated	219	D	6.731	female	76.2	diseased	1
A0002	treated	218	C	7.950	female	76.0	diseased	1
A0003	control	214	A	5.617	female	75.3	healthy	0
A0004	control	194	A	8.490	female	72.1	diseased	1
A0005	treated	185	D	7.127	female	72.6	healthy	0
A0006	treated	235	A	4.882	female	77.8	healthy	0

We specify logistic regression models in R using the `glm()` function. Just like in a linear model we have the outcome on the left hand side of the formula and the predictors on the right. We need to add an argument `family = binomial` to tell R to do logistic regression. We store the model in an object. Just like in linear regression...

```
mod_disease <- glm(status01 ~ treatment + age, family = binomial, data = dat)
```

We can look at a summary of the model output using `summary()`...

```
print(summary(mod_disease), digits = 3)
```

Call:

```
glm(formula = status01 ~ treatment + age, family = binomial,
    data = dat)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.847	-1.428	0.746	0.826	0.973

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-0.88412	1.23913	-0.71	0.476
treatmenttreated	-0.45345	0.21225	-2.14	0.033 *
age	0.01021	0.00589	1.74	0.083 .

```

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 587.20  on 499  degrees of freedom
Residual deviance: 579.67  on 497  degrees of freedom
AIC: 585.7

Number of Fisher Scoring iterations: 4

```

## odds ratios

Because we used some clever maths to convert the numbers from a linear model to a probability scale of zero to one the raw estimates from our model are actually 'log odds ratios'. In epidemiology we want to look at simple odds ratios. i.e.

$$\frac{\text{odds of outcome if have factor}}{\text{odds of outcome if dont have factor}}$$

So we get odds ratios by 'inverse logging' the raw estimates from the model. If we use the `get_model_data()` function from the `sjPlot` package it will automatically convert the estimates from a logistic regression model into odds ratios by inverse logging them...

```

get_model_data(mod_disease) %>%
  select(term:p.stars)

```

```

# A tibble: 2 x 8
  term          estimate std.error statistic p.value conf.low conf.high p.stars
<fct>         <dbl>     <dbl>     <dbl>   <dbl>   <dbl>   <dbl>   <chr>
1 treatmenttreated  0.635    0.212     -2.14  0.0326   0.419   0.963 *
2 age              1.01    0.00589    1.74  0.0827   0.999   1.02  ""

```

Compare the results with the raw estimates from `summary(mod_disease)` to see what's happened. For example, from the raw results -0.453 has become 0.635 as it has gone from `log(odds-ratio)` to `odds_ratio` etc.

## Interpreting the odds ratios

Odds ratios **multiply** when we interpret them. Let's look at what this means for categorical and numerical predictors...

### Categorical predictors

The odds ratio estimate here means 'how many times greater the odds of outcome are **if** the risk factor (etc) is present'. So for the treatment variable

(which can be control or treatment) the odds of disease if treated are 0.635 **times greater** than if untreated (control).

### Numerical predictors

And here they mean 'hw many times greater the odds of outcome are for **each unit change** in the variable'. So for the age variable the odds of disease are 1.01 **times greater** for each day older. So for 3 days older the odds of being diseased are  $1.01 \times 1.01 \times 1.01 \simeq 1.03$  greater.

### Things to watch out for

#### Factor levels

How does R know if you are predicting 'healthy' or 'diseased'?

#### Perfect predictors

E.g. all the males are diseased and all the females are healthy

#### Linear on logit

Disease risk might go up and then down