Chapter 15: Modelling proportion data using the binomial distribution

STATS 201/8

University of Auckland

Learning outcomes

In this chapter you will learn about:

- Binary (Bernoulli) data, odds and log-odds
- Modelling log-odds
- Modelling the response when it is binary (ungrouped data) via `glm`
- Modelling the response when it is binomial (grouped binary data) via `glm`
- Example 1: Space shuttle *Challenger* accident
- Example 2: Probability of retaining fish in a trawl
- Relevant `R`-code.

Section 15.1
Binary (Bernoulli) data, odds and log-odds

Binary (Bernoulli) data

Here we are considering the situation where the response can only take **two** possible values. These might be coded in the form of:

- Zeros or ones.
- `TRUE` or `FALSE`.
- `Yes` or `No`.
- `Success` or `Failure`..
- Or any other pair of categorical values.

This is called a binary response and can be modelled using the **Bernoulli** distribution.

A Bernoulli distribution is just the special case of the binomial distribution[1]

[1]Recall, the binomial models the number of “successes” out of a fixed number of Bernoulli trials.

Binary (Bernoulli) data...

Examples

A Bernoulli random variable is the outcome for a single trial expressed as a 0 or 1, E.g.,

- Whether or not I get a heads when I flip a coin.
- Whether or not I roll a six with a six-sided dice.
- Whether or not a soccer player scores a penalty kick.
- Whether or not I score a shot in basketball.
- Whether or not a green light is observed at a single set of traffic lights on my regular commute to work.
- Whether or not a patient survives an experimental procedure.

In each of these examples we get to choose which outcomes are assigned the values 0 or 1.

Typically, 0 = "No" (or "Failure"), and 1 = "Yes" (or "Success").

Bernoulli random variables

If Y is a Bernoulli random variable with parameter p, then Y will take the value 1 with probability p, and the value 0 with probability $1 - p$.

Since it is a probability, p must be a value that is between 0 and 1, i.e. $p \in [0, 1]$.

For Bernoulli trials, the usual terminology is to refer to Y as the number of successes, either **zero** or **one**, from a single trial.

It is easy to show[2] that the mean of a Bernoulli random variable is

$$\mathrm{E}[Y] = p$$

and that the variance is

$$\mathrm{Var}(Y) = p(1 - p)$$

[2]See STATS 210.

Odds

We will soon be modelling Bernoulli data using the `glm` function. To be able to make sense of the fitted model we first need to know about **odds**. The odds of an event occurring is given by

$$\text{Odds} = \frac{\text{probability event occurs}}{\text{probability \ event does not occur}} = \frac{\Pr(Y=1)}{\Pr(Y=0)} = \frac{p}{1-p}$$

Note that Odds must be a value between 0 to infinity, i.e. $\text{Odds} \in [0, \infty)$.

A little of bit of calculus gives us

$$p = \frac{\text{Odds}}{\text{Odds}+1}$$

So, if we know the odds of an event then we also know the probability of that event (and vice-versa).

What's the odds?

The definition of odds is given in the previous slide.

Luckily for us, the above definition of odds has exactly the same meaning that we give to the word "odds" when we use it in everyday speech when we use "odds" to describe how likely an event is to occur.

By way of example, let event A={I get an A- or better in STATS 201}.

- If the odds of A are 1 (i.e., 1-1, or even odds) then we are saying that A is as likely to occur as not. That is Pr(A)=0.5, and Pr(not A)=0.5.
- If the odds of A are 2 (i.e., 2-to-1) then we are saying that A is twice as likely to occur as not. That is Pr(A)=2/3 and Pr(not A)=1/3.
- If the odds of A are 0.25 (i.e., 0.25-to-1) then we are saying that A is only 1/4 as likely to occur as not. That is Pr(A)=1/5 and Pr(not A)=4/5.

Log-odds

The logarithm of the odds (the log-odds for short) is

$$\text{Log-Odds} = \log(\text{Odds}) = \log\left(\frac{p}{1-p}\right) \equiv \text{logit}(p)$$

Note that the log-odds (as a function of p) is called the *logit* function. Also, since $\text{Odds} \in [0, \infty)$ it follows that Log-Odds can take any value on the real line, i.e. $\text{Log-Odds} \in (-\infty, +\infty)$.

If we know the log-odds, then we can calculate p using the following:

$$p = \frac{\exp(\text{Log-Odds})}{1 + \exp(\text{Log-Odds})}$$

This is called the *logistic* function, and is well-known in mathematics as a function which maps the interval $(-\infty, +\infty)$ to $[0, 1]$. In `R` it is the `plogis`[3] function.

[3] `plogis` is so named because it calculates the probability distribution function of the logistic distribution. This is precisely the logistic function.

Section 15.2
Modelling log-odds

Why log-odds?

Our aim in this Chapter is to be able to fit models for binary data that allow the probability of success, p, to depend on explanatory variables. This is equivalent to allowing the odds or log-odds to depend on the explanatory variables.

For example, if x is a numeric explanatory variable then we have the following modelling options:

- $p = \beta_0 + \beta_1 x$
- $\text{Odds} = \beta_0 + \beta_1 x$
- $\text{Log-Odds} = \beta_0 + \beta_1 x$

Which would your suggest, and why?

Why log-odds?...

Since our model places no restrictions[4] on the values of β_0 or β_1, it is the case that $\beta_0 + \beta_1 x$ can take any value on the real line.

That is, $\beta_0 + \beta_1 x \in (-\infty, \infty)$.

- $p = \beta_0 + \beta_1 x$ ✗
  p must be between 0 and 1.
- Odds $= \beta_0 + \beta_1 x$ ✗
  Odds must be between 0 and infinity.
- Log-Odds $= \beta_0 + \beta_1 x$ ✓
  Log-Odds can be any real number.

[4] i.e., $\beta_0 \in (-\infty, \infty), \beta_1 \in (-\infty, \infty)$

Modelling log-odds

We shall apply our linear model to the log-odds, and so our general form of the model is:

$$\begin{aligned}\log(\text{Odds}) &= \text{logit}(p)\\ &= \text{logit}(\mu)\\ &= \text{logit}(E[Y|x_1\ldots])\\ &= \beta_0+\beta_1x_1+\ldots\end{aligned}$$

Remark: This is analogous to what we did when the response variable was Poisson count data. There we were using the model

$$\log(\mu)=\log(E[Y|x_1\ldots])=\beta_0+\beta_1x_1+\ldots$$

and note that $\log(\mu)\in(-\infty,\infty)$.

Modelling log-odds...

With a single numeric explanatory variable

When we have just a single explanatory variable x that is numeric (i.e., not a factor) then the linear model for log-odds is:

$$\log(\text{Odds}) = \beta_0 + \beta_1 x$$

In other words,

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x$$

where p is the probability of "success" for a subject with explanatory variable x.

This can be re-arranged in the logistic form

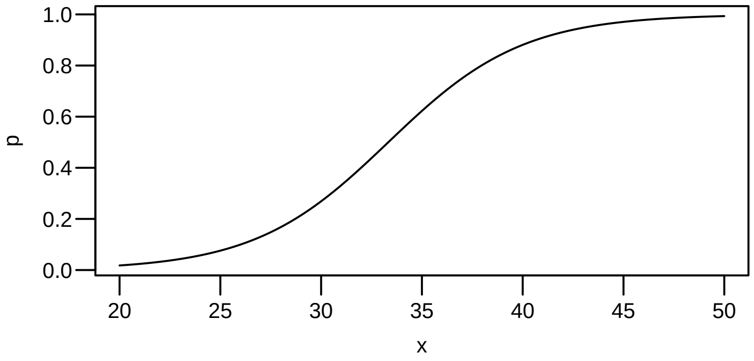
$$p = \frac{\exp(\beta_0 + \beta_1 x)}{1 + exp(\beta_0 + \beta_1 x)}$$

This equation forms an "S" shaped curve as a function of x.

Modelling log-odds. . .

With a single continuous explanatory variable. . .

For example, if $\beta_0 = -10$ and $\beta_1 = 0.3$ then the curve looks like:



- The greater the magnitude of β_1 the steeper the curve.
- If $\beta_1 < 0$ the curve is a reverse "S" shape.
- Changing β_0 changes the horizontal position of the curve.

Logistic regression

- The logistic function gives its name to this approach for modelling binary data – it is commonly called *logistic* regression.
- At the moment we are going to use logistic regression to model binary (*Bernoulli*) data, but it can also be used to model proportion data in the form of the number of "successes" divided by the number of trials. The number of "successes" is assumed to be binomially distributed, so logistic regression is often called a binomial GLM.
- In GLMs, the function that links μ to the linear predictor is called the *link* function. Here, $\mu = p$, and so the link function is the logit.

Section 15.3
Modelling the response when it is binary (ungrouped data) via `glm`

Example – Basketball

A few years ago the lecturer of an experimental design course conducted a basketball shooting experiment in class. In this experiment

- there were ten females and ten males;
- each shooter shot at the basket from 1m, 2m and 3m;
- each person took one shot from each distance;
- the order of the shooting distance was randomly chosen, and
- a 1 was recorded if the shot was successful, and a 0 was recorded if the shot was missed.

What do **you** think is the most important factor affecting the probability of making the shot in this experiment?

Inspect the data

Ungrouped format

The data are read in to dataframe `bb.df`. It contains 60 observations, since each of the 20 students takes 3 shots.

```
> bb.df = read.csv("Data/basketball.csv")
> head(bb.df, 10)
   distance gender basket
1         3      M      1
2         1      F      1
3         2      M      1
4         3      M      0
5         1      M      1
6         2      F      1
7         2      F      1
8         1      F      1
9         3      F      0
10        1      F      1
```

These data are in an *ungrouped* format as the response variable for each row is either a success or a failure (a 1 or a 0).

Basketball

Grouped format

Because the ungrouped responses are zeros and ones, it is hard to detect patterns in this kind of data. However we can group the data over each combination of gender and distance.

The xtabs function produces a cross-tabulation table[5]

```
> success.tbl = xtabs(basket~distance+gender, data = bb.df)
> success.tbl
        gender
distance  F  M
       1 10 10
       2  6  5
       3  2  1
```

Our conclusions should be kind of obvious from the table in this example, even without the model fitting.

[5] Also called a frequency table.

Basketball. . .

Logistic regression model formula

We want to fit a model that estimates how the probability of scoring is related to distance (numeric explanatory) and gender (factor explanatory). The interaction model (Chapter 8) is

$$\begin{aligned}\text{logit}(p_i) &= \log\left(\frac{p_i}{1-p_i}\right)\\ &= \beta_0 + \beta_1 distance_i + \beta_2 gender_i + \beta_3(gender_i \times distance_i)\end{aligned}$$

or in terms of p_i[6]

$$p_i = \frac{exp(\beta_0 + \beta_1 distance_i + \beta_2 gender_i + \beta_3(gender_i \times distance_i))}{1 + exp(\beta_0 + \beta_1 distance_i + \beta_2 gender_i + \beta_3(gender_i \times distance_i))}$$

with $Y_i \sim \text{Bernoulli}(p_i)$ and $gender_i$ is an indicator variable which is 0 if the shooter is female, and 1 if the shooter is male.

[6]We do not recommend that you try to write it this way in a test or assignment!

Basketball. . .

Fit the model. . .

Fitting the logistic regression model is easy.

We simply tell the glm function that the responses are Bernoulli random variables by setting family = binomial.[7]

```
> bb.fit = glm(basket ~ distance * gender, family = binomial,
+              data = bb.df)
```

Note that, as with Poisson GLMs, we **do not** transform the responses.

Aside: By default the glm() function uses the logit link function (i.e., does logistic regression) when we set family = binomial. Other choices are possible – see STATS 730.

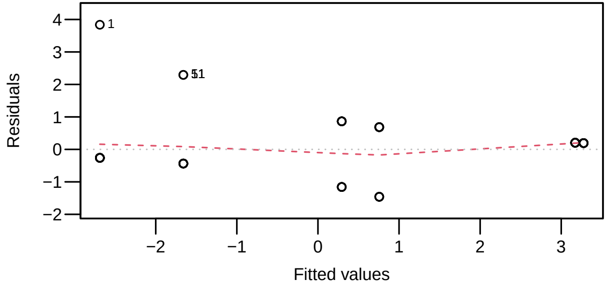
[7]Recall, the Bernoulli distribution is a special case of the binomial distribution.

Basketball. . .

Check the model. . .

Model checking is difficult when the data are *ungrouped*.[8]

```
> plot(bb.fit, which = 1, lty=2)
```



The plot of the residuals versus the fitted values is not particularly informative. Even if the model is appropriate, it can looked quite patterned. Influence checks are also of little use.

[8] This is because the data are *sparse*, in the sense that they are either 0's or 1's.

Basketball. . .

Inspect the fitted model

```
> summary(bb.fit)
```

```
Call:
glm(formula = basket ~ distance * gender, family = binomial,

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)        5.5878     1.9050   2.933  0.00335 **
distance          -2.4159     0.8181  -2.953  0.00314 **
genderM            0.6710     2.9235   0.230  0.81847
distance:genderM  -0.5668     1.3213  -0.429  0.66794
---
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 82.108  on 59  degrees of freedom
Residual deviance: 46.202  on 56  degrees of freedom
```

When the data are *ungrouped* we also **cannot** use the residual deviance in the same way we did for Poisson GLMs.

We just have to presume that the fitted model satisfies assumptions.

Basketball. . .

Simplify the model

There is no evidence of an interaction between gender and distance. We'll apply Occam's razor and drop it from the model.

```
> bb.fit1 = glm(basket ~ distance + gender, family = binomial, data = bb.df)
> summary(bb.fit1)
```

```
Call:
glm(formula = basket ~ distance + gender, family = binomial,

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   6.1469     1.5242   4.033 5.51e-05 ***
distance     -2.6648     0.6364  -4.188 2.82e-05 ***
genderM      -0.5478     0.7486  -0.732    0.464
---
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 82.108  on 59  degrees of freedom
Residual deviance: 46.392  on 57  degrees of freedom
```

Looks like we can drop gender, too.

Basketball. . .

Simplify the model. . .

```
> bb.fit2 = glm(basket ~ distance, family = binomial, data = bb.df)
> summary(bb.fit2)
```

```
Call:
glm(formula = basket ~ distance, family = binomial, data = bb.df)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   5.7980     1.4038   4.130 3.63e-05 ***
distance     -2.6310     0.6274  -4.193 2.75e-05 ***
---
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 82.108  on 59  degrees of freedom
Residual deviance: 46.937  on 58  degrees of freedom
```

Basketball. . .

Interpretting the fitted model

So, our final model is of the form

$$\text{Log-Odds} = \beta_0 + \beta_1 \times \textit{distance}$$

This means that a 1 metre increase in distance increases the log-odds by β_1.

In terms of odds, we have

$$\begin{aligned}\text{Odds} &= \exp(\beta_0 + \beta_1 \times \textit{distance}) \\ &= \exp(\beta_0) \times \exp(\beta_1)^{\textit{distance}}\end{aligned}$$

This means that a 1 metre increase in distance multiplies the odds by $\exp(\beta_1)$.

Basketball. . .

Interpretting the fitted model. . .

```
> coef(bb.fit2)
(Intercept)    distance
   5.797968   -2.631033
```

The estimated coefficient on distance, $\hat{\beta}_1$, is -2.63. This says that the *log-odds of success* decreases by 2.63 for every 1 metre increase of the shooter from the goal.

```
> exp(coef(bb.fit2))
 (Intercept)     distance
329.62899018   0.07200401
> 100*(1-exp(coef(bb.fit2)))
(Intercept)    distance
-32862.8990     92.7996
```

By exponentiating $\hat{\beta}_1$ we can now say that a 1 metre increase in the distance results multiplies the odds of shooting a basket by 0.072. However, it would be more natural to say that it results in a $100 \times (1 - 0.072)\% = 92.8\%$ reduction in the odds.

Basketball. . .

Interpretting the fitted model. . .

For our Executive Summaries we need confidence intervals rather than point estimates:

```
> (bb.ci2 = confint(bb.fit2))
Waiting for profiling to be done...
                2.5 %    97.5 %
(Intercept)  3.422396  9.037020
distance    -4.103523 -1.568945
> 100*(1-exp(bb.ci2))
                  2.5 %        97.5 %
(Intercept) -2964.27509 -840767.86136
distance       98.34856      79.17351
```

A 1 metre increase in the distance results in a reduction in the odds of scoring of between 79.2% and 98.3%

Estimating p

Basketball example

How do we estimate the probability of a successful shot from distances of 1, 2 and 3 m?

By default, `predict` will give estimates on the log-odds scale, that is, it will calculate the linear predictor $\hat{\beta}_0 + \hat{\beta}_1 \times \textit{distance}$.

```
> predn.df=data.frame(distance = 1:3)
> bb.logit.pred = predict(bb.fit2, newdata = predn.df)
> bb.logit.pred
        1          2          3 
 3.1669343  0.5359009 -2.0951325 
```

We can easily convert this to the response (i.e., probability) scale

```
> plogis(bb.logit.pred)
        1         2         3 
0.9595708 0.6308584 0.1095708 
```

Even easier is to use the `type="response"` argument of `predict`:

```
> predict(bb.fit2, newdata = predn.df, type="response")
        1         2         3 
0.9595708 0.6308584 0.1095708 
```

Confidence Intervals for p

Basketball example. . .

Confidence intervals need to first be calculated on the log-odds scale, and then transformed back to the probability scale.

```
> bb.logit.predses = predict(bb.fit2, newdata = predn.df, se.fit = TRUE)$se.fit
> bb.logit.predses
        1         2         3 
0.8151018 0.3812977 0.6432312 
> # Lower and upper bounds of CIs for the log-odds
> lower = bb.logit.pred - 1.96*bb.logit.predses
> upper = bb.logit.pred + 1.96*bb.logit.predses
> ci = cbind(lower, upper)
```

and convert these to probabilities:

```
> plogis(ci)
       lower     upper
1 0.82768876 0.9915452
2 0.44733541 0.7830016
3 0.03370361 0.3027157
```

We see that the probability of a goal from 1 metre distance is between 0.828 and 0.991, but drops to between 0.034 and 0.303 at 3 metres.

Confidence Intervals for p the easy way

Basketball example...

Or we can use the `predictGLM` function from the `s20x` package.

To get confidence intervals for log-odds

```
> predictGLM(bb.fit2,newdata = data.frame(distance = 1:3),type="link")
***Estimates and CIs are on the link scale***
         fit        lwr        upr
1  3.1669343  1.5693642  4.7645045
2  0.5359009 -0.2114289  1.2832308
3 -2.0951325 -3.3558424 -0.8344225
```

or to get confidence intervals for the probabilities

```
> predictGLM(bb.fit2,newdata = data.frame(distance = 1:3),type="response")
***Estimates and CIs are on the response scale***
        fit        lwr       upr
1 0.9595708 0.82769294 0.9915450
2 0.6308584 0.44733881 0.7829992
3 0.1095708 0.03370437 0.3027108
```

In the next section, we reproduce the above analysis after first grouping the data.

Section 15.4
Modelling the response when it is binomial (grouped binary data) via glm

The binomial distribution

A binomial random variable is the number of successes that occur over a fixed number (n) of Bernoulli trials, all with the same probability of success (p):

- The number of heads if I flip a coin fifty times.
- The number of sixes I get if I roll four dice.
- The number of penalties a football team scores in a penalty shoot-out.
- The number of successful basketball shots out of ten attempts.
- The number of green lights out of the total number of traffic lights on my regular commute to work.
- The number of patients who survive an experimental procedure, out of the number that underwent that procedure.
- The number of O-rings that fail when a space shuttle is launched, out of the total of six O-rings on the solid fuel rockets.

One difference between a binomial and Poisson random variable is that the binomial has an upper limit set by the number of trials.

Example revisited – Basketball: Grouped data

In many situations it is possible to format the binary data so that they are *grouped*.

With grouped data, all trials with the same values of the explanatory variables are aggregated in the same row, along with the number of “successes” and “failures”.

We demonstrate with our basketball data:

```
> #Load dplyr package to manipulate data frames
> library(dplyr)
> bb.grouped.df = bb.df %>% group_by(gender,distance) %>%
+                          summarize(n=n(),propn=sum(basket)/n)
> #Change tibble back to a data frame
> bb.grouped.df = data.frame(bb.grouped.df)
> bb.grouped.df
  gender distance  n propn
1      F        1 10   1.0
2      F        2 10   0.6
3      F        3 10   0.2
4      M        1 10   1.0
5      M        2 10   0.5
6      M        3 10   0.1
```

Basketball: Grouped data

We can still use `glm` to fit a logistic regression model to grouped data.

Previously, we considered each of the 60 attempts to score as a separate Bernoulli trial.

Instead, we now consider the number of successes out of 10 attempts at each level of gender and distance. This is a binomial random variable.

Both approaches will give you the same estimates, confidence intervals, and conclusions, but there are some advantages to having grouped data. Especially, being better able to check model assumptions.

Basketball: Grouped data...

Fitting the model

We set the proportion of successes as the response, and use the argument `weights` to specify the number of trials associated with each observation.

```
> bb.fit3 = glm(propn ~ distance * gender, weights = n,
+                 family = binomial, data = bb.grouped.df)
> summary(bb.fit3)
```

```
Call:
glm(formula = propn ~ distance * gender, family = binomial, data = bb.grouped.df,

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)        5.5878     1.9050   2.933  0.00335 **
distance          -2.4159     0.8181  -2.953  0.00314 **
genderM            0.6710     2.9236   0.230  0.81848
distance:genderM  -0.5668     1.3214  -0.429  0.66795
---
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 38.2749  on 5  degrees of freedom
Residual deviance:  2.3688  on 2  degrees of freedom
```

Basketball: Grouped data...

Checking the model...

The grouped data output is equivalent to that from working with the *ungrouped* data, except that the deviance values have changed.

We may now interpret residual plots and check the residual deviance statistic, notwithstanding that we ideally require the expected count of both successes and failures to be reasonably large, say > 5.[9]

```
> 1 - pchisq(2.3688, 2)
[1] 0.3059297
```

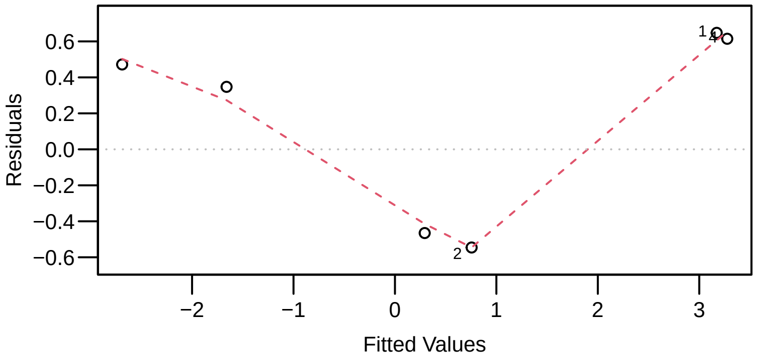
No problems with the residual deviance.

[9] This is not satisfied for the basketball data, but we will proceed anyway.

Basketball: Grouped data...

Checking the model...

```
> plot(bb.fit3, which = 1, lty=2)
```



We only have six observations (one for each combination of distance and gender), so it is difficult to see if there is a pattern in the residuals, but in this case none are large enough to worry us.
The fitted model is not perfect, but should still be useful.

An alternative way to specify the model

Basketball: Grouped data. . .

Sometimes the grouped data are provided as the number of successes and failures:

```
> bb.grouped.df = transform(bb.grouped.df, success=n*propn, fail=n*(1-propn))
> bb.grouped.df
  gender distance  n propn success fail
1      F        1 10   1.0      10    0
2      F        2 10   0.6       6    4
3      F        3 10   0.2       2    8
4      M        1 10   1.0      10    0
5      M        2 10   0.5       5    5
6      M        3 10   0.1       1    9
```

An alternative way to specify the model...

Basketball: Grouped data...

Then, on the left-hand side of the model formula we provide `glm` with the names of the two columns containing the number of successes and failures, and omit the `weights` argument.

```
> bb.fit4 = glm(cbind(success, fail) ~ distance * gender, family = binomial,
+     data = bb.grouped.df)
> summary(bb.fit4)
```

```
Call:
glm(formula = cbind(success, fail) ~ distance * gender, family = binomial,

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)        5.5878     1.9050   2.933  0.00335 **
distance          -2.4159     0.8181  -2.953  0.00314 **
genderM            0.6710     2.9236   0.230  0.81848
distance:genderM  -0.5668     1.3214  -0.429  0.66795
---
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 38.2749  on 5  degrees of freedom
Residual deviance:  2.3688  on 2  degrees of freedom
```

Basketball: Grouped data. . .

Model selection

In practice, we would simplify these models by removing the interaction and gender effects in turn, as we did for analysis of the ungrouped data.

We have not done so because the results are unchanged.

Section 15.5
Example 1: Space shuttle Challenger accident

Space shuttle *Challenger* accident

This example tells a sad story showing what can happen if standard linear models are mis-applied to proportion data.

The NASA space shuttle *Challenger* broke up during launch on the cold morning of 28 January 1986. Most of the crew survived the initial break-up, but are believed to have been killed when the crew capsule hit the ocean at high speed.

At the time, it was the most expensive human accident that had ever occurred.[10]

It was particularly traumatic for the American people, because the shuttle was carrying the first civilian astronaut, Christa McAuliffe, who was a high-school social studies teacher. Approximately 17% of Americans were watching the launch live.

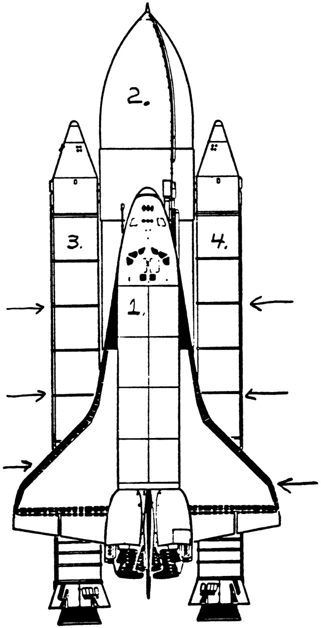
[10] Unfortunately, an even more deadly and expensive accident occurred just months later - Chernobyl.

Space shuttle *Challenger* accident...

Subsequent investigation found that O-ring failures were the cause of the disaster.

The temperature[a] at the launch site was a chilly $31^\circ F$. The risk of O-ring failure in cold weather had been grossly underestimated due to using a linear model on proportion data.

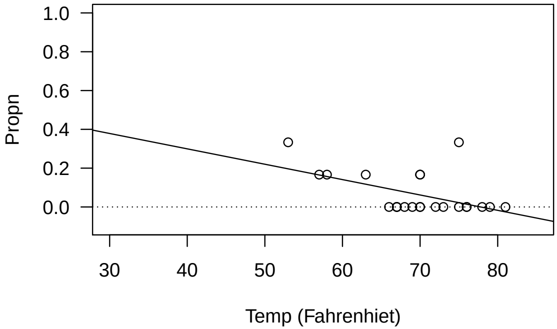
[a] This is a fraction of a a degree Celsius below freezing point.



Space shuttle *Challenger* accident...

The space shuttle solid-fuel rockets have a total of 6 O-rings. It was suspected that O-ring reliability was influenced by temperature.

The rockets retrieved from previous launches were examined for O-ring distress, and the proportion of distressed O-rings was plotted against temperature and a simple linear regression was fitted:



The simple linear regression estimates a negative probability of O-ring distress for temperatures above about $79^\circ F$!?!

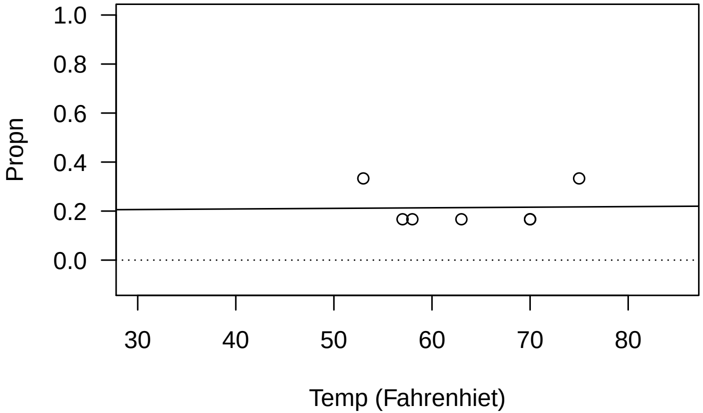
Space shuttle *Challenger* accident. . .

To fix the problem with the negative estimated probabilities, "statisticians" at NASA decided to remove all the zero data values since they felt that the zero values contained no information about the probability of O-ring distress.

This kind of stupidity occurs even today.

Space shuttle *Challenger* accident. . .

With the zero values removed there is no evidence of a relationship between temperature and O-ring distress,



and so it was decided to approve the launch on that $31^\circ F$ morning.

Space shuttle *Challenger* accident. . .

73 seconds after lift-off the shuttle blew apart:

These data should have been analysed using a binomial GLM that is appropriate for proportion data.

Space shuttle *Challenger* accident. . .

Logistic regression model

```
> Space.df = read.table("Data/ChallengerShuttle.txt", head = TRUE)
> Space.df$Temp
 [1] 66 70 69 68 67 72 73 70 57 63 70 78 67 53 67 75 70 81 76 79 75 76 58
> Space.df$Failure
 [1] 0 1 0 0 0 0 0 0 1 1 1 0 0 2 0 0 0 0 0 0 2 0 1
> Space.gfit=glm(cbind(Failure, 6-Failure)~Temp, family = binomial,
+                 data = Space.df)
```

```
> summary(Space.gfit)
```

```
Call:
glm(formula = cbind(Failure, 6 - Failure) ~ Temp, family = binomial,

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.08498    3.05247   1.666   0.0957 .
Temp        -0.11560    0.04702  -2.458   0.0140 *
---
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 24.230  on 22  degrees of freedom
Residual deviance: 18.086  on 21  degrees of freedom
```

Space shuttle *Challenger* accident...

Conclusions[11]

Our CI for the probability of an O-ring failing at $31^\circ F$ is

```
> predictGLM(Space.gfit, newdata = data.frame(Temp=31), type = "response")
***Estimates and CIs are on the response scale***
        fit       lwr       upr
1 0.8177744 0.1596025 0.9906582
```

The rockets have 6 O-rings, so the expected number of O-rings failures is

```
> 6*predictGLM(Space.gfit, newdata = data.frame(Temp=31), type = "response")
***Estimates and CIs are on the response scale***
       fit      lwr      upr
1 4.906646 0.957615 5.943949
```

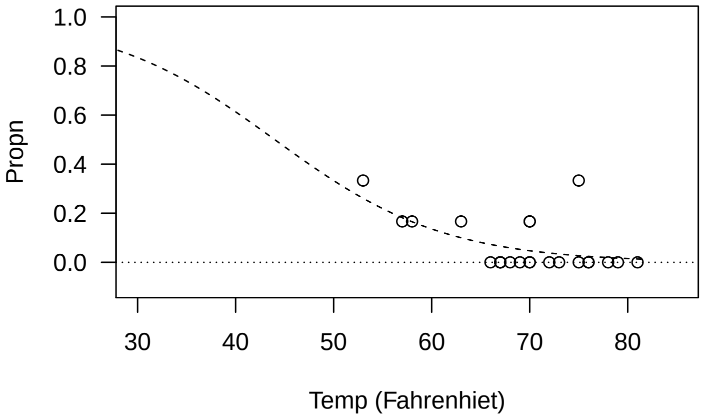
Note that this is quite a wide confidence interval because we are estimating the probability at a temperature that is well beyond those observed in the dataset. This is called "extrapolation", and is always risky.

The real message here is that there **could** be a very high probability of disaster.

[11] Model checks are left as an exercise.

Space shuttle *Challenger* accident...

The fit of this generalized linear model to the O-ring data looks like:



This model predicts that the probability of an O-ring experiencing distress at $31^\circ F$ is 0.818. This corresponds to expecting $6 \times 0.818 = 4.91$ distressed O-rings.

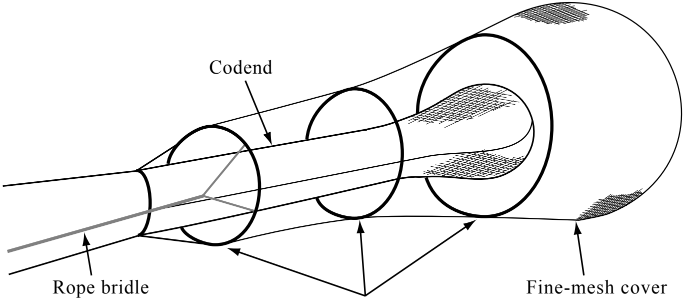
Section 15.6
Example 2: Catching the right size fish

Example – Fishing

In commercial fishing it is important to let small fish escape so they can grow and breed. Experiments are frequently done with trawl gear to determine how the probability of retaining a fish depends on its size.

The experiment consisted of observing the number of fish (at given fork lengths) entering a trawl codend, and the number of those retained by it.

In the dataframe `Haddock.df`, `codend` is the number in the codend and `cover` is the number that escape the codend and are retained in the cover. The total number of haddock is therefore `codend + cover`.



Fishing

```
> Haddock.df = read.table("Data/Haddock.dat", head = TRUE)
> Haddock.df = transform(Haddock.df,propn=codend/(codend+cover))
> head(Haddock.df, 17)
   forklen codend cover      propn
1     19.5      0     2 0.00000000
2     20.5      0     5 0.00000000
3     21.5      0    11 0.00000000
4     22.5      0    28 0.00000000
5     23.5      1    53 0.01851852
6     24.5      5    46 0.09803922
7     25.5      1    35 0.02777778
8     26.5      3    27 0.10000000
9     27.5      1     5 0.16666667
10    28.5      0     3 0.00000000
11    29.5      1     2 0.33333333
12    30.5      0     0        NaN
13    31.5      0     2 0.00000000
14    32.5      3     2 0.60000000
15    33.5      4     4 0.50000000
16    34.5      5    12 0.29411765
17    35.5      8     9 0.47058824
```

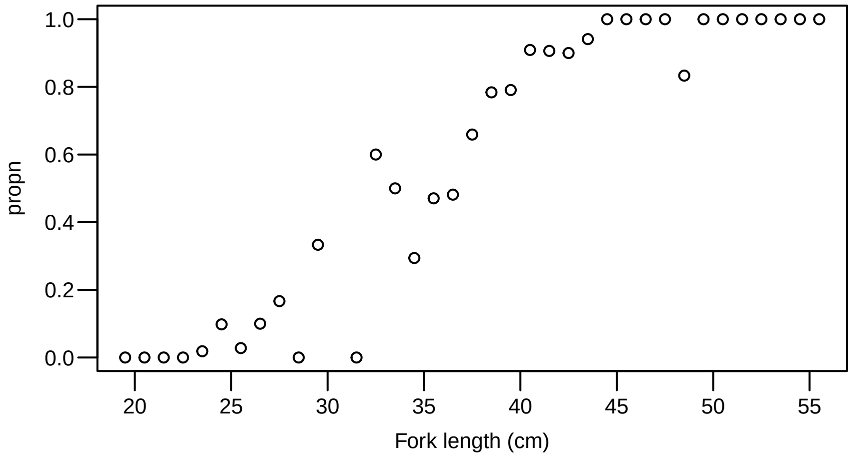
Fishing...

```
> tail(Haddock.df, 20)
   forklen codend cover     propn
18    36.5     13    14 0.4814815
19    37.5     29    15 0.6590909
20    38.5     29     8 0.7837838
21    39.5     34     9 0.7906977
22    40.5     30     3 0.9090909
23    41.5     29     3 0.9062500
24    42.5     18     2 0.9000000
25    43.5     16     1 0.9411765
26    44.5     11     0 1.0000000
27    45.5     12     0 1.0000000
28    46.5      9     0 1.0000000
29    47.5      3     0 1.0000000
30    48.5      5     1 0.8333333
31    49.5      3     0 1.0000000
32    50.5      5     0 1.0000000
33    51.5      2     0 1.0000000
34    52.5      2     0 1.0000000
35    53.5      1     0 1.0000000
36    54.5      1     0 1.0000000
37    55.5      4     0 1.0000000
```

Fishing. . .

Plot the data

```
> plot(propn ~ forklen, data = Haddock.df, xlab = "Fork length (cm)")
```



Note that the proportions retained seem to follow an "S" shape.

Fishing. . .
Fitting the model

Let's fit a logistic regression model.

Note that the data are grouped, and the number of "successes" and "failures" are given by the variables codend and cover, respectively[12].

```
> Haddock.glm = glm(cbind(codend,cover) ~ forklen, family = binomial,
+                   data = Haddock.df)
```

[12] Retaining the fish may by a success for the fisherman, but perhaps not for the fish!

Fishing. . .

Modelling proportion data using log-odds

```
> summary(Haddock.glm)
```

```
Call:
glm(formula = cbind(codend, cover) ~ forklen, family = binomial,

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -10.63219    0.86468  -12.30   <2e-16 ***
forklen       0.30396    0.02363   12.86   <2e-16 ***
---
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 432.464  on 35  degrees of freedom
Residual deviance:  23.436  on 34  degrees of freedom
```

Fishing. . .

Checking the fitted model

As with the Poisson case, we need to check the residual deviance by comparing it to a χ^2 distribution. We can do this here as the data are grouped: we have many observations for each level of fork-length, so the data are not ‘sparse’.

The residual deviance is 23.44 on 34 degrees of freedom. The *P*-value is

```
> 1-pchisq(23.44,34)
[1] 0.9132025
```

which indicates no significant problems with the fitted model.

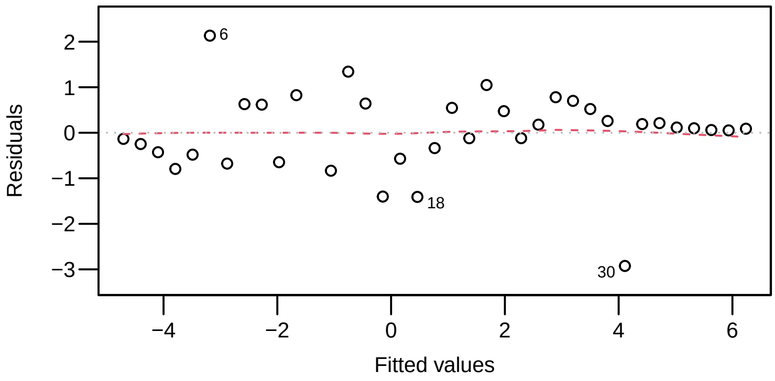
Recall: If the residual deviance indicated lack of fit then we would have to refit using `family = quasibinomial`.

Fishing. . .

Checking the fitted model. . .

Let's check the residual plot

```
> plot(Haddock.glm,which=1, lty=2)
```



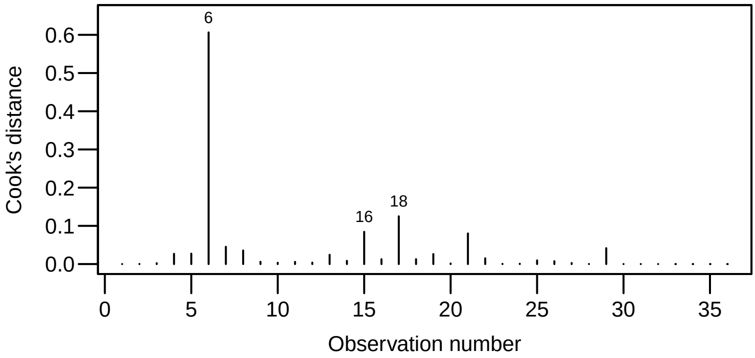
Looks good, other than one largish residual corresponding to the 48.5 cm fish that was in the cover.

Fishing...

Checking the fitted model...

Lets check the influence plot

```
> plot(Haddock.glm,which=4)
```



Hmmm, the fit is somewhat influenced by the retention of small fish of length 24.5 cm. However, this is known to happen with trawl gear, so this observation will not be removed.

Fishing...

Interpretation

```
>   summary(Haddock.glm)$coef
               Estimate Std. Error   z value     Pr(>|z|)
(Intercept) -10.6321889 0.86467598 -12.29615 9.499049e-35
forklen       0.3039569 0.02363082  12.86273 7.295469e-38
```

Remember, the linear model is on the log-odds scale. So, the value 0.304 corresponds to an estimated increase in the log-odds of codend retention for every one cm increase in `forklen`.

The 95% confidence intervals for β_0 and β_1 are:

```
>   confint(Haddock.glm)
Waiting for profiling to be done...
                  2.5 %     97.5 %
(Intercept) -12.4563256 -9.0501022
forklen       0.2604867  0.3535532
```

Fishing. . .

Interpretation. . .

Exponentiating the above confidence intervals gives:

```
> exp(confint(Haddock.glm))
Waiting for profiling to be done...
                   2.5 %      97.5 %
(Intercept) 3.893019e-06 0.000117379
forklen     1.297561e+00 1.424118770
```

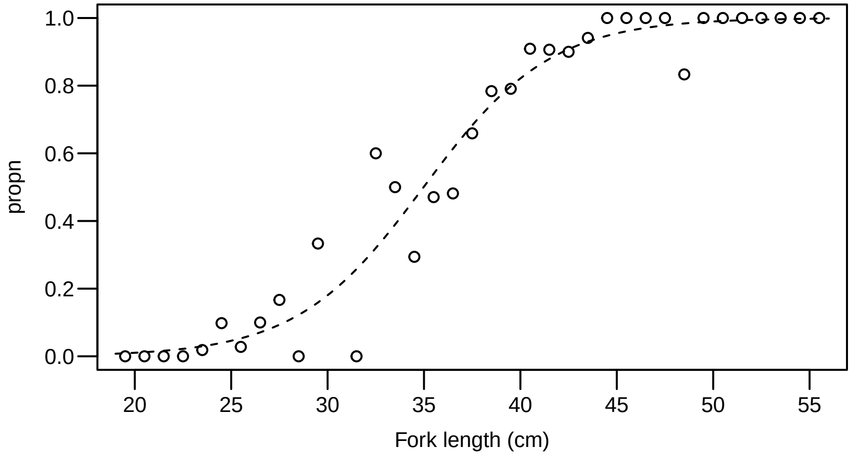
In our **Executive Summary** we could say something like

"Every 1 cm increase in the fork length of a haddock multiplies the odds of it being retained in the codend by between 1.3 and 1.42".

Or, "Every 1 cm increase in the fork length of a haddock corresponds to an increase in odds of it being retained in the codend of between 30% and 42%".

Fishing. . .

Here is a plot of the data with the fitted “S” curve overlaid:



Fishing. . .

Confidence intervals for p

To estimate the probability of retention for haddock of lengths 25, 35 and 45 cm:

```
> predn=predictGLM(Haddock.glm, data.frame(forklen=c(25,35,45)), type="response")
***Estimates and CIs are on the response scale***
> predn
         fit        lwr        upr
1 0.04594542 0.02639853 0.07879451
2 0.50157550 0.43868954 0.56441166
3 0.95460393 0.92887749 0.97131200
```

The probability that a 25 cm haddock is retained in the codend is between 0.026 and 0.079. This increases to between 0.439 and 0.564 for 35 cm haddock, and between 0.929 and 0.971 for 45 cm haddock.

Section 15.7
Relevant R-code.

Most of the R-code you need for this chapter

Ungrouped data

Ungrouped data are binary i.e., a 0 or 1 response value for each observation (for example `bb.df`). Fit the model with `glm` using `family=binomial`:

```
> bb.fit2 = glm(basket ~ distance, family = binomial, data = bb.df)
```

These type of data are known as sparse (because either the count of successes is 0 or the count of failures is 0) and we cannot test our model assumptions effectively as we would like to. If possible we should group our data.

Most of the R-code you need for this chapter...

Grouped data

We were also able to analyse the basketball data as grouped observations which allows us to check assumptions including if we have to make a `family= quasibinomial` adjustment to our model.

Syntax 1 - specify proportion of successes and number of trials

```
> bb.fit3 = glm(propn ~ distance * gender,weight=n,
+              family = binomial,data = bb.grouped.df)
```

Syntax 2 - specify frequency of successes and failures

```
> bb.grouped.df = transform(bb.grouped.df, success=n*propn, fail=n*(1-propn))
> bb.fit4 = glm(cbind(success, fail) ~ distance * gender,
+              family = binomial,data = bb.grouped.df)
```

Most of the R-code you need for this chapter. . .

Both model syntaxs give exactly the same results and we can perform the residual deviance check as follows:

```
> 1-pchisq(bb.fit4$deviance,bb.fit4$df.residual)
> # or directly
> 1 - pchisq(2.3688, 2)
```

If the p-value is below 0.05 then we have evidence against the assumed binomial distribution and need to refit our model using `family=quasibinomial`.

Most of the R-code you need for this chapter. . .

Confidence intervals can be calculated for the parameters and back-transformed and interpreted as a multiplicative effect on the odds (or as a % change if you prefer).

```
> (bb.ci2 = confint(bb.fit2))
> exp(bb.ci2)
```

Confidence intervals for probabilities take a bit more effort since they require calcuating a confidence interval on the logit scale (i.e., for log-odds) and then backtransforming using the logistic function.

Fortunately, we can use `predictGLM`.

```
> bb.pred.intervals = predictGLM(bb.fit2,
+                               newdata = data.frame(distance = c(1, 2, 3)),
+                               type="response")
***Estimates and CIs are on the response scale***
> bb.pred.intervals
        fit        lwr       upr
1 0.9595708 0.82769294 0.9915450
2 0.6308584 0.44733881 0.7829992
3 0.1095708 0.03370437 0.3027108
```