Moving beyond linearity in response

UBCO MDS — DATA 570

Generalized Linear Models

- Linear Models and the need for more flexibility
- Poisson regression
- Binary logistic regression

Linear Models and the Need for a More General Approach

A linear model is a predictive model where the expected value of the response or outcome variable can be expressed as a linear combination of predictor variables.

Noise or error is modelled by adding it to the expected value.

Normally, if the noise is modelled at all, it is modelled as a normal random variable with mean 0.

Usually, linear models can be fit using least-squares methods, where a linear system of equations must be solved in order to find the parameter estimates.

Examples: simple and multiple linear regression; autoregressive time series models For count data and many other kinds of data, normality is not realistic.

How Generalized Linear Models Differ from Linear Models

A generalized linear model is a predictive model where the expected value of the response variable is a *function* of a linear combination of predictor variables.

Noise or error is modelled with specific distributions.

Count data is better modelled with binomial, Poisson, or negative binomial, and so on.

If continuous measurements are always positive, such as time until failure, other models, such as Weibull or lognormal are appropriate.

Generalized linear models are usually fit using maximum likelihood estimation or a related method.

Motivated data

Cigarette butts collected during the 2012 International Coastal Cleanup in Oregon. Thomas Jones / Ocean Conservancy

Cigarette butts are the largest source of ocean trash

Poisson Regression

The *cigbutts* data set (in the *MPV* package) gives counts of cigarette butts at locations along a sidewalk as a function of distance from a smoking gazebo.

We can use Poisson regression to model this count data as a function of distance.

Poisson Distribution

Recall: the Poisson distribution with mean λ is given by

$$P(X = x) = \frac{e^{-\lambda}\lambda^x}{x!}.$$

A common approach to fitting the Poisson distribution regression is to include a predictor x through a *log-linear* relationship:

$$\log(\lambda) = \beta_0 + \beta_1 x.$$

The log function used in this way is referred to as the *link* function since it links the distribution parameter λ with the predictor x.

This link function is preferred here because it recognizes that λ must be positive, while the right hand side of the above equation can take any real value.

Likelihood for the cigarette butts data

If X_i is the number of butts observed at the ith location, its expected value would be λ_i and the corresponding distance might be denoted as d_i.

The model for the ith observation is really

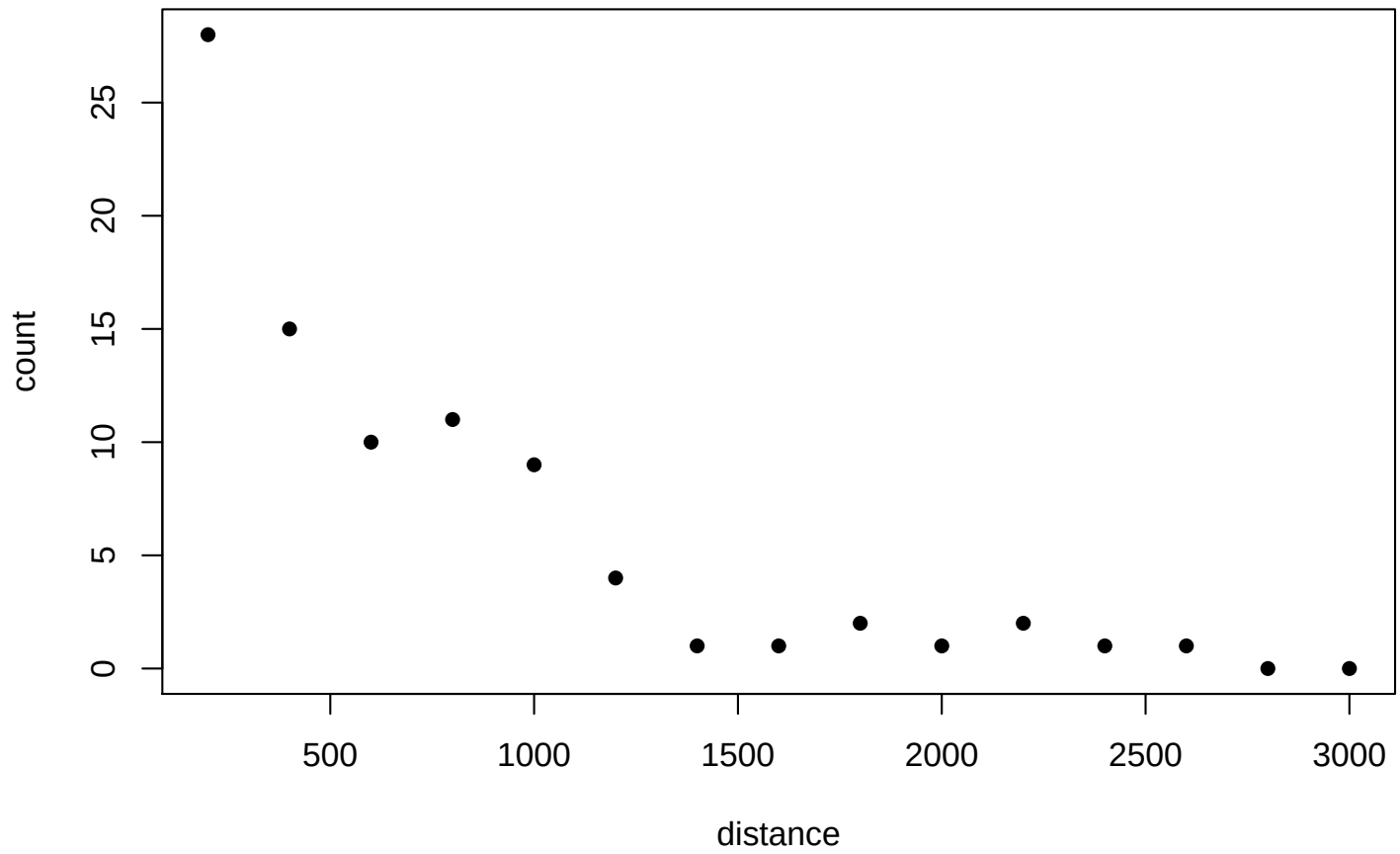
$$P(X_i = x_i) = \frac{e^{-\lambda_i}\lambda_i^{x_i}}{x_i!}.$$

The likelihood function is the product of such probabilities

The log likelihood is then the sum of the logs of the probabilities:

$$\log L = -\sum_{i=1}^{n} \lambda_i + \sum_{i=1}^{n} x_i \log \lambda_i - \sum_{i=1}^{n} \log x_i!.$$

Visualizing the Cigarette Butt Counts



Scatterplot of log of cigarette butt counts versus log of distance.

Predicting Counts Using Distance

We suspect a linear relation between the log of λ_i and d_i:

$$\log(\lambda_i) = \beta_0 + \beta_1 d_i$$

Let's suppose we know that $\beta_0 = 3.55$. Then we could say that

$$\log(\lambda_i) = 3.55 + \beta_1 d_i$$

Plugging this into the log likelihood expression gives

$$\log L = -\sum_{i=1}^{n} e^{3.55+\beta_1 d_i} + \sum_{i=1}^{n} x_i(3.55 + \beta_1 d_i) - \sum_{i=1}^{n} \log x_i!.$$

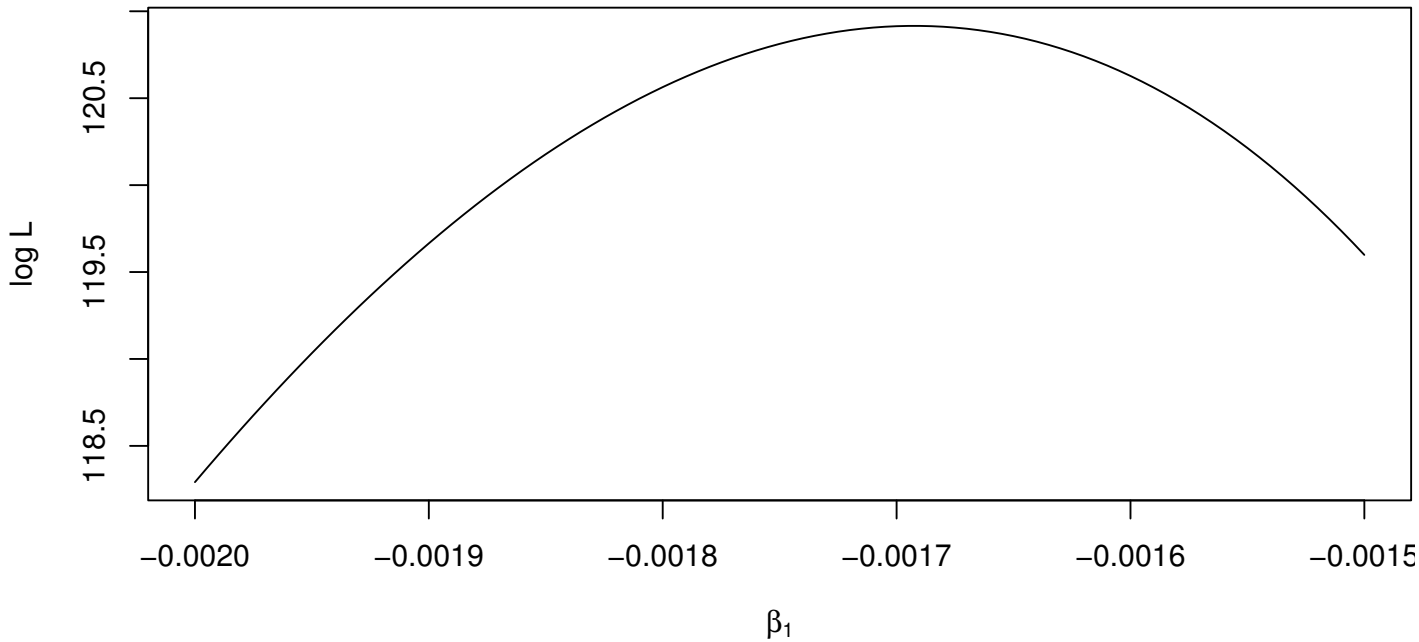
The x_i's are the observed counts and the d_i's are the distances, so we can plot this as a function of β.

Predicting Counts Using Distance

Log likelihood function:

```
x <- cigbutts$count; d <- cigbutts$distance; n <- length(x)
b <- seq(-.002, -.0015, length = 101)
logLterms <- outer(b, 1:n,
   function(b, i) -exp(3.55+ b*d[i]) + x[i]*(3.55+b*d[i]))
logL <- apply(logLterms, 1, sum)
par(mar=c(4,4, .1, .1))
plot(b, logL, xlab = expression(beta[1]), ylab = "log L",
 type = "l")
```

Predicting Counts Using Distance



Log likelihood curve for the cigarette butts example. The maximizer is near -.0017.

A predictive model for log of the expected number of cigarette butts would be

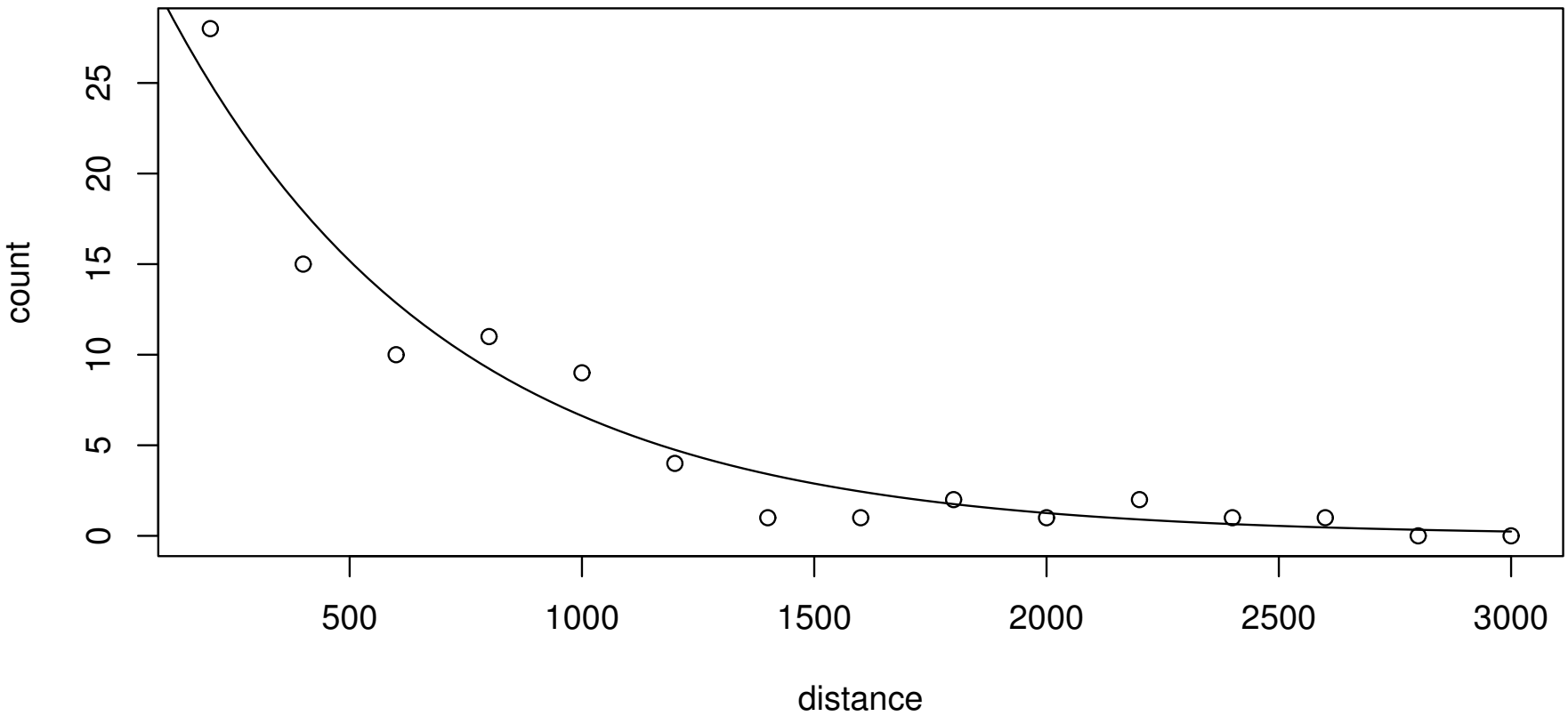
$$\widehat{\log(\lambda)} = 3.55 - .0017d$$

Using Built-In Software

The glm() function fits Poisson regressions and logistic regressions fairly straightforwardly.

```
cig.glm <- glm(x ~ d, family = poisson)

coef(cig.glm)

## (Intercept) d

## 3.553514 -0.001696

par(mar=c(4, 4, .1, .1))

plot(count ~ distance, data = cigbutts)

curve(exp(3.55 - .00166*x), 0, 3000, add = TRUE)
```

Fitted curve



Additional output

```
summary(cig.glm)$coefficients

#                Estimate   Std. Error  z value     Pr(>|z|)

#(Intercept)  3.553514259 0.1735102026 20.48015 3.236877e-93

#distance    -0.001695636 0.0002008638 -8.44172 3.127010e-17
```

Note, in particular, the standard errors of the parameter estimates.

Simulating from the Fitted Model

Recall that we simulate *n* Poisson random numbers using *rpois(n, lambda)*.

If we have a vector of *n* predictor values in x, and a fitted model of the form

$$\log(\lambda) = \beta_0 + \beta_1 x$$

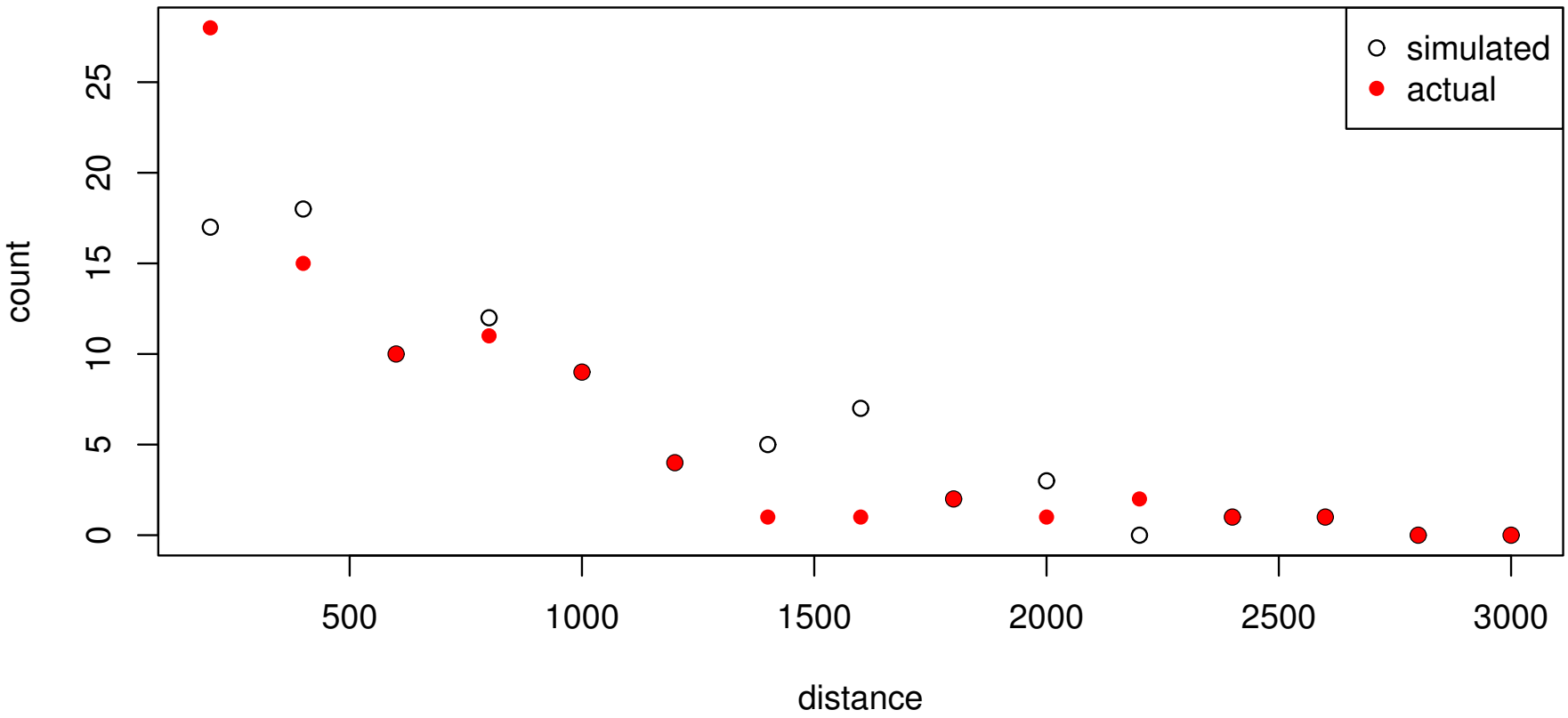
we can simulated n corresponding Poisson responses, using

$$\lambda = e^{\beta_0 + \beta_1 x}.$$

To simulate the numbers of cigarette butts as a function of distance, use

Simulating from the Fitted Model

```
lambda <- exp(3.55 - 0.00166*d)

n <- nrow(cigbutts) # No. of observations in the data set

simcounts <- rpois(n, lambda = lambda)

simcigbutts <- data.frame(count = simcounts, distance = d)

plot(count ~ distance, data = simcigbutts,

    ylim = range(simcigbutts$count, cigbutts$count))

points(count ~ distance, data = cigbutts, col="red", pch=16)

legend("topright", legend = c("simulated", "actual"),

pch=c(1, 16), col=c("black", "red"))
```


simulated
actual
count
0
5
10
15
20
25
500
1000
1500
2000
2500
3000
distance

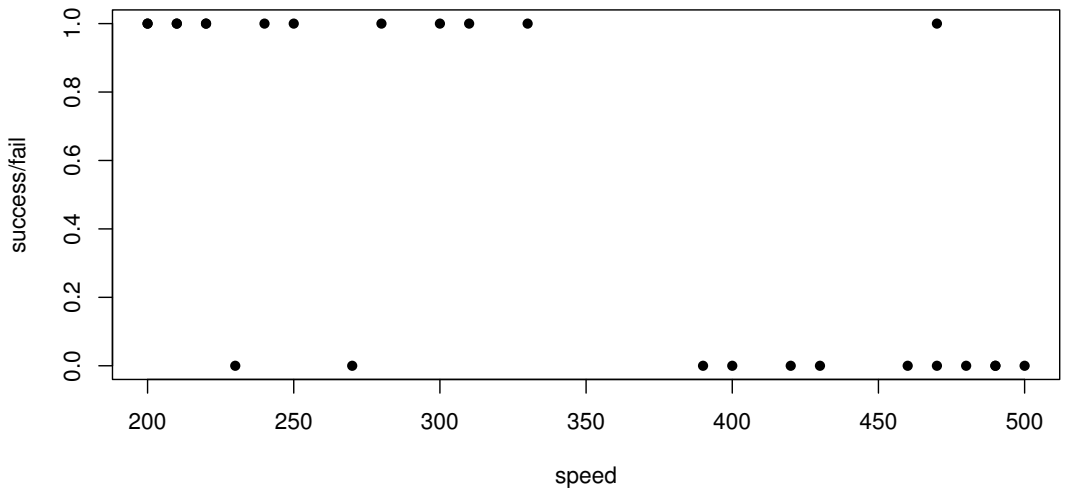
Modelling Binary Responses

The data in p13.1 in the *MPV* package describes successes and failures of surface-to-air missiles as they relate to target speed.

```
library(MPV)
plot(p13.1, xlab = "speed", ylab = "success/fail", pch=16)
```

Surface-to-air missile successes (1) and failures (0) as they relate to target speed (in knots).

Modelling Binary Responses



The first observation to make is that fitting a straight line to such data makes no sense, since the plotted points do not at all scatter about such a line.

Furthermore, if such a line were to be fit to the data, it would necessarily take values outside the interval $[0, 1]$ on subsets of the domain; interpretation of such values would be difficult.

Modelling Binary Responses

In fact, the preferred interpretation of output arising from the fitting of models to such data is that of probability.

That is, useful models can provide answers to questions such as, "What is the probability of success at a given target speed?"

Since probabilities must lie within the interval $[0, 1]$, we must consider models based on nonlinear functions.

There are many functions which have values in $[0, 1]$.

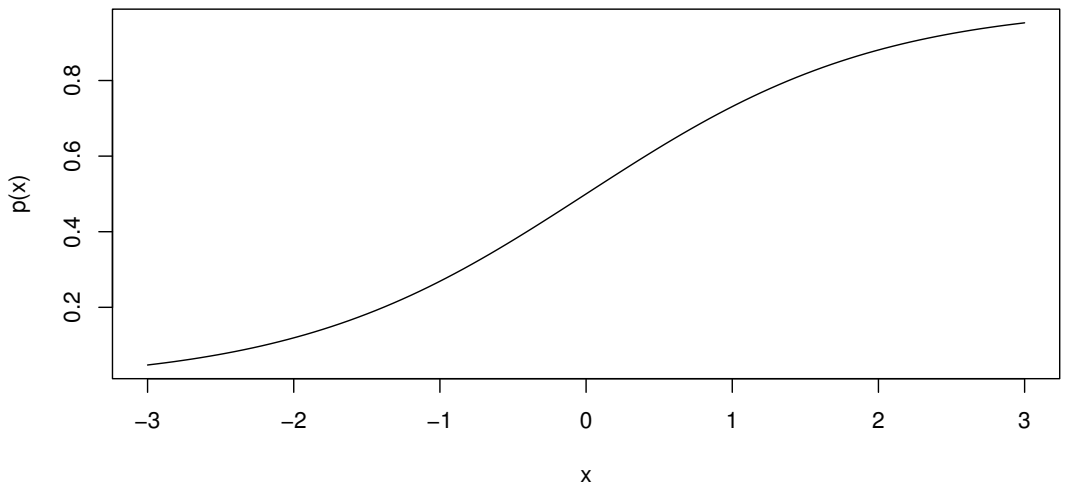
For the current example, we might reasonably believe that the probability of success decreases as target speed increases.

Modelling Binary Responses

Perhaps the most popular function for this purpose is the *logistic* function

$$p(x) = \frac{e^x}{e^x + 1}.$$

```
curve(exp(x)/(1 + exp(x)), from = -3, to = 3, ylab="p(x)")
```



Modelling Binary Responses

A bit of algebra allows us to express x in terms of p, yielding the *logit* function:

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

While p is restricted to take values between 0 and 1, the logit function can take any possible value, so relating the logit function to a straight line or other linear combination is a possibility. For example,

$$\text{logit}(p(x)) = \beta_0 + \beta_1 x$$

which means that we can express the probability of an event in terms of a covariate x, using a linear function, but the probability is related to the linear function through the logit.

Modelling Binary Responses

The logit is an example of a *link function*, since it links the expected response, in this case the probability $p(x)$ to the linear function of the covariate(s).

To fit the logistic regression model to the missile success data, try

```
p13.glm <- glm(y ~ x, data = p13.1, family = binomial)
```

Note that we did not specify the link function; the default choice with the binomial family is the logit.

Modelling Binary Responses

```
coef(p13.glm)

#(Intercept)           x

# 6.0708839  -0.0177047
```

The Coefficient part of the output tells us that the logit of the probability of success as a linear function of target speed has intercept 6.07 and slope -.0177.

Modelling Binary Responses

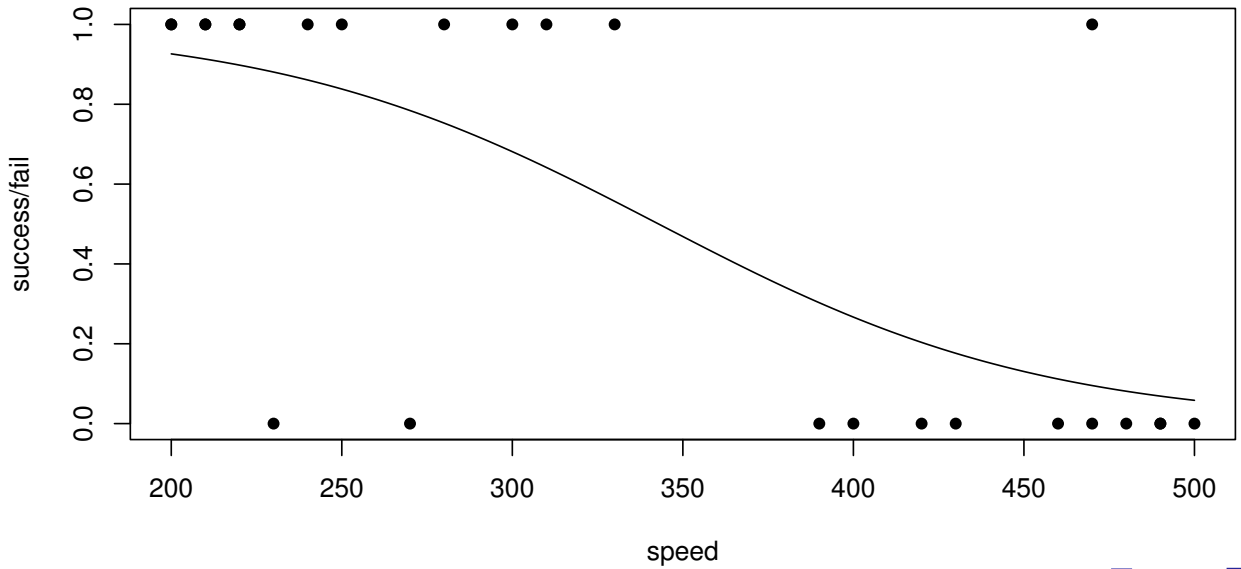
More output:

```
summary(p13.glm)$coefficients
#               Estimate  Std. Error   z value     Pr(>|z|)
#(Intercept)  6.0708839 2.108996265  2.878566 0.003994883
#x           -0.0177047 0.006075513 -2.914107 0.003567073
```

Standard error estimates for these parameter estimates are supplied and indicate, in particular, that the slope is clearly negative.

Modelling Binary Responses - Visualizing the Model

```
plot(p13.1, xlab = "speed", ylab = "success/fail", pch=16)
newspeeds <- 200:500 # predict at these target speeds
lines(newspeeds, predict(p13.glm,
  newdata=data.frame(x = newspeeds),  type = "response"))
```



Simulating from the Fitted Model

Recall that we simulate n Bernoulli random numbers using rbinom(n, 1, p).

If we have a vector of n predictor values in x, and a fitted model of the form

$$\text{logit}(p) = \beta_0 + \beta_1 x$$

we can simulated n corresponding Bernoulli responses, using

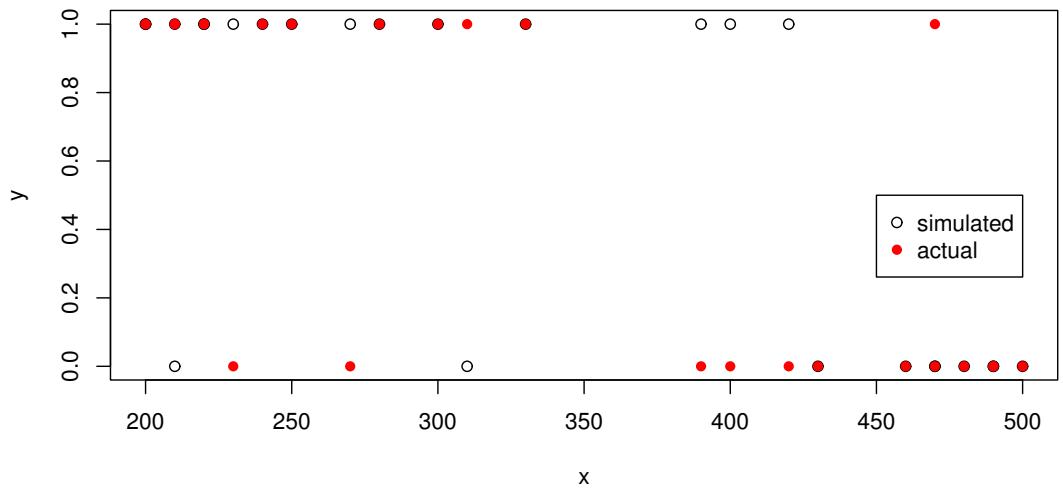
$$p = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}.$$

To simulate the numbers of missile successes as a function of speed, use

```
p <- exp(6.0709-0.0177*p13.1$x)/(1+exp(6.0709-0.0177*p13.1$x))
n <- nrow(p13.1) # No. of observations in the data set
simy <- rbinom(n, 1, prob = p)
```

Simulating from the Fitted Model

```
simp13.1 <- data.frame(y = simy, x = p13.1$x)

plot(y ~ x, data = simp13.1)

points(y ~ x, data = p13.1,  col="grey", pch=16)

legend(450, .5, legend = c("simulated", "actual"), pch=c(1, 16),

    col=c("black", "grey"))
```



What to Take Away from this Segment

Generalized Linear Models

- the concept: fitting predictive models for the mean response using maximum likelihood estimation
- understanding the concept of a link function
- understanding the logit function
- fitting Poisson regression and logistic regression with glm()
- simulating from the fitted models