Generalized linear models and logistic regression

1 Introduction

- Our data are pairs of predictors and response $(X_1, Y_1), \ldots, (X_n, Y_n)$ as usual. Now, however, we assume that the response is categorical and more specifically binary, i.e. $Y \in \{0, 1\}$. [The predictors can be categorical, continuous, or some of each.] This is known as binary regression or **binary classification**.
- As usual we are interested in estimating the conditional mean function $m(x) = \mathbb{E}[Y_i|X_i = x]$. When Y is binary this is the same as $\mathbb{P}(Y_i = 1|X_i = x)$ and so it must always be between 0 and 1. If we modeled this probability as a linear function in x, that is

$$\mathbb{P}(Y = 1|X = x) = \beta_0 + \sum_{j=1}^{p} \beta_j x_j, \tag{1}$$

we could just run linear regression as usual. But in general linear functions do not stay between 0 and 1 and so we tend to think (1) is an unrealistic model for the conditional mean.
- Instead of the linear model (1) we will assume that there is a function g for which

$$\mathbb{P}(Y = 1|X = x) = g^{-1}\Big(\beta_0 + \sum_{j=1}^{p} \beta_j x_j\Big) \iff g\Big(\mathbb{P}(Y = 1|X = x)\Big) = \beta_0 + \sum_{j=1}^{p} \beta_j x_j. \tag{2}$$

The function g is refered to as a **link function**. By picking g to have range $[0, 1]$, we can ensure the conditional mean is always a valid probability. If we choose the identity link function $g(m) = m$, we recover (1), and so we see that (2) generalizes the linear model (1). For this reason it is an example of a **generalized linear model** (GLM).
- The motivation behind GLMs is to incorporate non-linearity into our model in a way that allows for the same kind of activities – namely, parameter estimation, prediction, inference, and diagnostics – as in linear regression.
- We start by discussing some relevant link functions and other examples of GLMs. Then we talk parameter estimation, confidence intervals, and prediction.

2 Generalized linear models

- To specify the model in (2) we need to pick a link function g with range in $[0, 1]$. We mention two common examples, one more common than the other.
- The **probit** link is the inverse normal CDF,

$$g(m) := \Phi^{-1}(m) \iff g^{-1}(t) = \Phi(t) \tag{3}$$

The **logit** link is

$$g(m) := \log\Big(\frac{m}{1-m}\Big) \iff g^{-1}(t) = \frac{\exp(t)}{1 + \exp(t)}. \tag{4}$$

Regression with the probit link is called probit regression. Regression with the logit link is called logistic regression (since the inverse link is the logistic function). Plotting the probit and logit links shows that they are similar and so either regression will give similar results.

- The probit model is correct if we believe Y was obtained by thresholding responses in the linear model. That is, if

$$\begin{aligned} Z_i &= \beta_0 + \sum_{j=1}^p \beta_j X_{ij} + \epsilon_i, \epsilon_i \sim N(0, \sigma^2) \\ Y_i &= \mathbf{1}\{Z_i > c\}, \end{aligned}$$

  then

$$m(x) = \mathbb{P}(Y_i = 1 | X_i = x) = \mathbb{P}(Z_i > c | X_i = x) = 1 - \Phi\left(c - \frac{\beta_0 + \sum_{j=1}^p \beta_j X_{ij}}{\sigma}\right) = \Phi\left(\frac{\beta_0 + \sum_{j=1}^p \beta_j X_{ij} - c}{\sigma}\right).$$

  The logistic model does not have as close a connection to the linear model. But it is easier to fit.

- There are many other examples of useful GLMs. Two concrete ones are Binomial regression and Poisson regression, both of which are good models for count data, where $Y \in \{0, 1, 2, \dots\}$
  - In Binomial regression we suppose $Y \sim \text{Bin}(N, p(x))$ where N is known and $p(x) = g^{-1}(\beta^\top x)$. Since p should be between 0 and 1, either the probit or logit links are good choices for g.
  - In Poisson regression we suppose $Y \sim \text{Pois}(\lambda(x))$ where $\lambda(x) = g^{-1}(\beta^\top x)$. The most common choice of link is the exponential function $g(\lambda) = \exp(\lambda)$.
- For the most part we will focus on logistic regression with binary outcomes.

3 Maximum Likelihood Estimation for GLMs

- Now we want to use data $(x_1, y_1), \dots, (x_n, y_n)$ to fit the parameters $\hat{\beta}$ in a GLM. What is typical, and what we will cover now, is estimating β by maximizing a likelihood. This is especially convenient because there is nice software and theory for maximizing the likelihood and computing confidence intervals, hypothesis tests, etc.

  That being said, it is worth mentioning that maximum likelihood is not the only approach. In principle there are many ways to estimate β. For instance we could have used the method of least squares and minimized

$$\sum_{i=1}^n (y_i - g^{-1}(\beta_0 + \sum_{j=1}^p \beta_j x_{ij}))^2.$$

  Nevertheless for GLMs maximum likelihood is every bit the foundational method that OLS is for LMs. In fact Example 1 below shows that the OLS estimator is the MLE when the model is Normal.

- To define the maximum likelihood estimator (MLE) we need a likelihood. Let $f(y, x; \theta)$ be the conditional density of $Y|X = x$ of the response for a given set of parameters θ. The **likelihood** of the observed data $y_1, \dots, y_n$ conditional on $X_1 = x_1, \dots, X_n = x_n$ is

$$\prod_{i=1}^n f(y_i, x_i; \theta). \tag{5}$$

  Since logarithm is a monotonic function the likelihood is maximized at the same value of θ as the log-likelihood:

$$\ell_n(\theta) = \sum_{i=1}^n \log f(y_i, x_i; \theta) \tag{6}$$

  To maximize the likelihood we set the gradient of the log-likelihood $\dot{\ell}_n(\theta) := \nabla_\theta \ell_n(\theta)$ – also known as the **score function** – equal to 0. For the types of problems we will consider the likelihood is strongly concave, so there is a unique maximum. Hence the **maximum likelihood estimator** (MLE) $\hat{\theta}_{\text{MLE}}$ is the unique value of parameters satisfying

$$\dot{\ell}_n(\hat{\theta}_{\text{MLE}}) = 0. \tag{7}$$

- **Example 1**: In our usual linear model with Normal errors $Y_i|X_i = x \sim N(\beta_0 + \sum_{j=1}^p \beta_j x_j, \sigma^2)$. The parameters are $\theta = (\beta_0, \ldots, \beta_p, \sigma^2)$. The conditional density of $Y_i|X_i = x$ is

$$f(y_i, x_i; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\bigg(-(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2/2\sigma^2\bigg). \tag{8}$$

So the log-likelihood is

$$\ell_n(\theta) = \frac{-1}{2\sigma^2} \sum_{i=1}^n \Big(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij}\Big)^2 - \frac{n}{2}\log\sigma^2 + const. \tag{9}$$

Taking partial derivatives with respect to $\beta_0, \ldots, \beta_p, \sigma^2$ and setting them equal to 0 leads to the equations:

$$\begin{aligned} \mathbf{X}^\top Y &= (\mathbf{X}^\top\mathbf{X})\hat\beta_{\text{MLE}} \\ n &= \frac{1}{\hat\sigma^2_{\text{MLE}}} \sum_{i=1}^n (y_i - \hat\beta_0 - \sum_{j=1}^p \hat\beta_j)^2, \end{aligned} \tag{10}$$

which can be rearranged to read

$$\begin{aligned} \hat\beta_{\text{MLE}} &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top Y, \\ \hat\sigma^2_{\text{MLE}} &= \frac{1}{n}\sum_{i=1}^n (y_i - \hat\beta_0 - \sum_{j=1}^p \hat\beta_j)^2. \end{aligned}$$

Thus the MLE for the intercept and coefficients is the same as our usual least squares estimator. The MLE for σ^2 is the sum of the squared residuals divided by n, which we know is a biased estimator of σ^2. This is typical of the MLE: it is attractive because it is a (nearly) automatic rule for estimating parameters, but is usually worse than the best possible estimator for a given problem.

- **Example 2**: Logistic regression: $Y_i|X_i = x \sim \text{Bern}(m_i(\beta))$ where

$$m_i(\beta) = \frac{\exp(\beta_0 + \sum_{j=1}^p \beta_j x_{ij})}{1 + \exp(\beta_0 + \sum_{j=1}^p \beta_j x_{ij})}.$$

The parameters are $\theta = \beta$ – unlike in linear regression, there is no extra nuisance parameter for the errors. The conditional distribution of $Y_i|X_i = x$ is

$$f(y_i, x_i; \beta) = [m_i(\beta)]^{y_i}[1 - m_i(\beta)]^{(1-y_i)}. \tag{11}$$

The log-likelihood can be written as

$$\begin{aligned} \ell_n(\beta) &= \sum_{i=1}^n y_i \log(m_i(\beta)) + (1 - y_i)\log(1 - m_i(\beta)) \\ &= \sum_{i=1}^n y_i(\beta_0 + \sum_{j=1}^p \beta_j x_{ij}) + \log(1 - m_i(\beta)) \end{aligned} \tag{12}$$

In taking the gradient of ℓ_n we will use the fact that

$$\nabla m_i(\beta) = m_i(\beta)(1 - m_i(\beta))(1 x_i)^\top.$$

From this, we can work out that the score function, i.e. the gradient of ℓ_n with respect to β, is

$$\dot\ell_n(\beta) = \sum_{i=1}^n y_i(1\ x_i)^\top - m_i(\beta)(1\ x_i)^\top.$$

Setting the score equal to 0, and rewriting in matrix notation, we have the non-linear system of equations

$$\mathbf{X}^\top Y = \mathbf{X}^\top \vec{m}(\hat{\beta}_{\text{MLE}}) \tag{13}$$

where $\vec{m}(\beta) = (m_1(\beta), \ldots, m_n(\beta))$. As a sanity check, in the case of linear regression, $\vec{m}(\beta) = \mathbf{X}\beta$, and we recover the normal equations for OLS.

- For logistic regression (and many other cases) the MLE does not have a closed form solution. Instead we maximize the likelihood using an iterative algorithm. The most common approach is a variant of Newton's method called **Fisher scoring**: given a current iterate β_t, update

$$\beta_{t+1} = \beta_t + [-\ddot{\ell}(\beta_t)]^{-1}\dot{\ell}(\beta_t). \tag{14}$$

  Remember that $\dot{\ell}(\beta)$ is a $p \times 1$ vector representing the gradient of the log-likelihood. The **Hessian** $\ddot{\ell}(\beta)$ is a $p \times p$ matrix of 2nd derivatives:

$$[\ddot{\ell}(\beta)]_{ik} = \frac{\partial^2 \ell}{\partial \beta_i \beta_j}.$$

  Each update in (14) can be recast as a weighted least squares problem which makes computing the updates particularly straightforward.

4 Sampling distribution and confidence intervals

- In this section I will write θ^* for the true value of the parameters. Typically in GLMs we do not have an exact expression for the sampling distribution of $\hat{\theta}_{\text{MLE}}$ about θ^*. Instead, the best we can do is appeal to asymptotics: under suitable regularity conditions, as $n \to \infty$:

$$\hat{\theta}_{\text{MLE}} | X_1, \ldots, X_n \to N(\theta^*, [I(\theta^*)]^{-1}). \tag{15}$$

  The convergence implied by (15) is convergence in distribution.

  From (15) we see that the MLE, which is not in general unbiased, is at least asymptotically unbiased for the true parameters The asymptotic variance depends on the inverse of the **Fisher information**. Fisher information is a function which takes as input a potential value of the parameters θ, and outputs the expectation of the Hessian of the log-likelihood at a given value of the parameters:

$$I(\theta) := \mathbb{E}_\theta[-\ddot{\ell}_n(\theta) | X_1, \ldots, X_n]$$

  $I(\theta^*)$ is the Fisher information evaluated at the (unknown) true parameters θ^*.

  I will not give a rigorous derivation of (15), but I've included a breezy description of why it should be true at the end of this section.

- We cannot directly use the asymptotic sampling distribution in (15) for inference as the inverse information $[I(\theta^*)]^{-1}$ depends on the unknown true parameters θ^*. Instead we can estimate the information by plugging in $\hat{\theta}_{\text{MLE}}$ for θ, and use the estimated covariance $[I(\hat{\theta}_{\text{MLE}})]^{-1}$ to construct a confidence interval.

- **Example**: Logistic regression. Remember that in logistic regression the parameters are just $\theta = \beta$. We have already calculated the score function (gradient of the log-likelihood) for logistic regression:

$$\dot{\ell}_n(\beta) = \sum_{i=1}^n y_i x_i^\top - m_i(\beta) x_i^\top.$$

  Taking another round of partial derivatives gives the Hessian matrix $\ddot{\ell}_n(\beta)$:

$$-\ddot{\ell}_n(\beta) = \sum_{i=1}^n x_i x_i^\top m_i(\beta)[1 - m_i(\beta)] =: \mathbf{X}^\top \mathbf{V}_\beta \mathbf{X},$$

where $\mathbf{V}_\beta$ is the $n \times n$ diagonal matrix with i diagonal entry $(\mathbf{V}_\beta)_{ii} = m_i(\beta)[1 - m_i(\beta)]$. Thus from (15) we have that

$$\hat{\beta}_{\text{MLE}}|X_1, \ldots, X_n \to N(\beta^*, (\mathbf{X}^\top \mathbf{V}_{\beta^*} \mathbf{X})^{-1}).$$

This is somewhat interpretable, as it looks like the sampling distribution of a weighted least squares estimator, with weights $(\mathbf{V}_\beta)_{ii} = m_i(\beta)[1 - m_i(\beta)]$. The weights are largest when $m_i(\beta) = \frac{1}{2}$, whereas zero weight is given to observations where $m_i(\beta) = 0$ or $m_i(\beta) = 1$.

Plugging $\hat{\beta}$ for β^* in the estimated variance, our confidence interval for β_j^* is

$$C_j := \hat{\beta}_j \pm z^{(1-\alpha/2)} \sqrt{[\mathbf{X}^\top \mathbf{V}_{\hat{\beta}} \mathbf{X}]^{-1}_{j+1,j+1}}$$

This is an asymptotic $(1 - \alpha)$ confidence interval for β_j.

- C_j is an asymptotic $(1 - \alpha)$ confidence interval for β_j. In general, confidence intervals constructed using plug-in estimates of the Fisher information will typically have asymptotically correct coverage if the generalized linear model is correct. However with small number of samples there are other methods that give more accurate confidence intervals. The default implementation in R uses one of these other methods.

- Here is the breezy derivation of (15). Assume $\theta = \beta$, i.e. there are no nuisance parameters like σ^2. We begin by taking a 1st-order Taylor expansion of the score function around the true parameter β^*, and evaluating at $\hat{\beta}$:

$$\dot{\ell}_n(\hat{\beta}) \approx \dot{\ell}_n(\beta^*) + [\ddot{\ell}_n(\beta^*)](\hat{\beta} - \beta^*) \tag{16}$$

By definition $\dot{\ell}_n(\hat{\beta}) = 0$, and so we can rewrite this as

$$(\hat{\beta} - \beta^*) \approx [-\ddot{\ell}_n(\beta^*)]^{-1} \dot{\ell}_n(\beta^*) \tag{17}$$

The right most term $\dot{\ell}_n(\beta^*)$ is the score function evaluated at the true value of the parameters. This is a sum of n i.i.d. random vectors so asymptotically we believe it should be Normal with the appropriate mean and variance. First we show that it has zero mean. Taking a derivative with respect to coordinate j, we have

$$\begin{aligned} \mathbb{E}_{\beta^*}\left[\partial_j \log f(Y; \beta^*)\right] &= \mathbb{E}\left[\frac{\partial_j f(Y; \beta^*)}{f(Y; \beta^*)}\right] \\ &= \int \partial_j f(y; \beta^*)\, dy \\ &= \partial_j \int f(y; \beta^*)\, dy \\ &= \partial_j 1 = 0. \end{aligned} \tag{18}$$

To derive the variance we use two facts without proof: (a) $-\ddot{\ell}_n(\beta^*) \approx I(\beta^*)$ and (b) when evaluated at $\beta = \beta^*$, the variance of the score is equal to the information

$$\text{Var}_{\beta^*}[\dot{\ell}_n(\beta^*)] = I(\beta^*).$$

From this we conclude that

$$\text{Var}_{\beta^*}[\hat{\beta}] \approx [I(\beta^*)]^{-1} I(\beta^*) [I(\beta^*)]^{-1} = [I(\beta^*)]^{-1}. \tag{19}$$

5 Odds and odds ratio

- Interpreting the coefficients in a logistic regression is challenging. The easiest way to interpret the coefficients is in terms of odds and odds ratio.

- Logistic regression does linear modeling on the logit scale: that is,

$$\eta(x) := \log\left(\frac{m(x)}{1-m(x)}\right) = \beta_0 + \sum_{j=1}^{p} \beta_j x_j. \tag{20}$$

  The quantity $\text{odds}(x) = m(x)/(1-m(x))$ is called the **odds**. The long-run frequency based interpretation of odds is this: at a given x, if the response were sampled many times, we would expect to see $\text{odds}(x)$ values of $Y = 1$ for every one time we saw $Y = 0$.

  The odds ratio of x' versus x is $\text{odds}(x')/\text{odds}(x)$. This is a ratio of ratios. The most natural way to interpret the meaning of the coefficients β by seeing what happens to the odds ratio as x increases. For example, in a one-variable logistic regression where

$$\log\left(\frac{m(x)}{1-m(x)}\right) = \beta_0 + \beta_1 x_1,$$

  we have that the odds ratio is

$$\frac{\text{odds}(x_1+1)}{\text{odds}(x_1)} = \exp(\beta_1).$$

  This means that if x_1 is increased by one unit, we expect the ratio of how often $Y = 1$ over how often $Y = 0$ to multiply by a factor of $\exp(\beta_1)$.

6 Prediction with logistic regression

- Now suppose we observe a new predictor $X_{n+1} = x_{n+1}$, and would like to predict the probability that $Y_{n+1} = 1$. Our prediction for the log odds is

$$\hat{\eta}(x_{n+1}) = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_{n+1,j},$$

  hence our prediction for the probability that $Y_{n+1} = 1$ given $X_{n+1} = x_{n+1}$ is

$$\hat{m}(x_{n+1}) = \frac{\exp(\hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_{n+1,j})}{1 + \exp(\hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_{n+1,j})}.$$

O-rings data

(From Faraway, ELM Ch. 3.) "In January 1986, the space shuttle Challenger exploded shortly after launch. An investigation was launched into the cause of the crash and attention focused on the rubber O-ring seals in the solid rocket boosters. At lower temperatures, rubber becomes more brittle and is a less effective sealant. At the time of launch, the temperature was 31° F. Could the failure of the O-rings have been predicted? In the 23 previous shuttle missions for which data existed, some evidence of damage due to blow-by and erosion was recorded on some O-rings."

```
library(faraway)
```

```
## Warning: package 'faraway' was built under R version 4.3.2
```

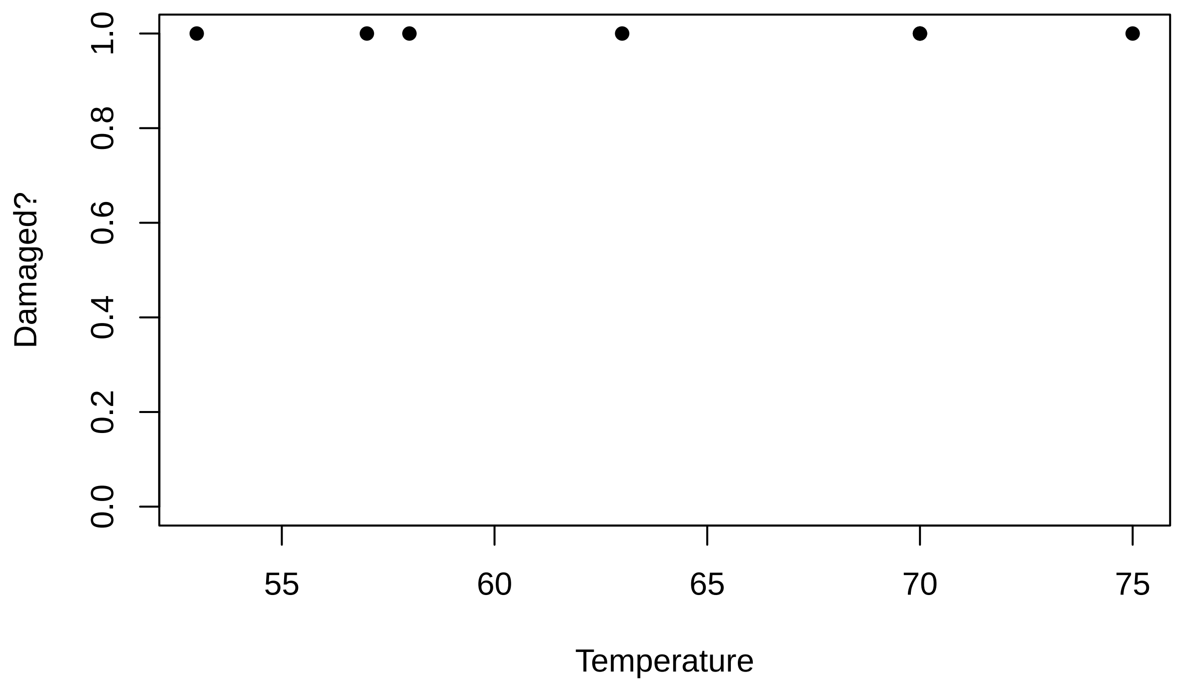
```
data(orings)
orings$damage = (orings$damage > 0)
attach(orings)
head(orings)
```

```
##   temp damage
## 1   53   TRUE
## 2   57   TRUE
## 3   58   TRUE
## 4   63   TRUE
## 5   66  FALSE
## 6   67  FALSE
```

A case study in the importance of good EDA

We plot the data as NASA did: by focusing only on the distribution of temperature when O-rings were truly damaged.

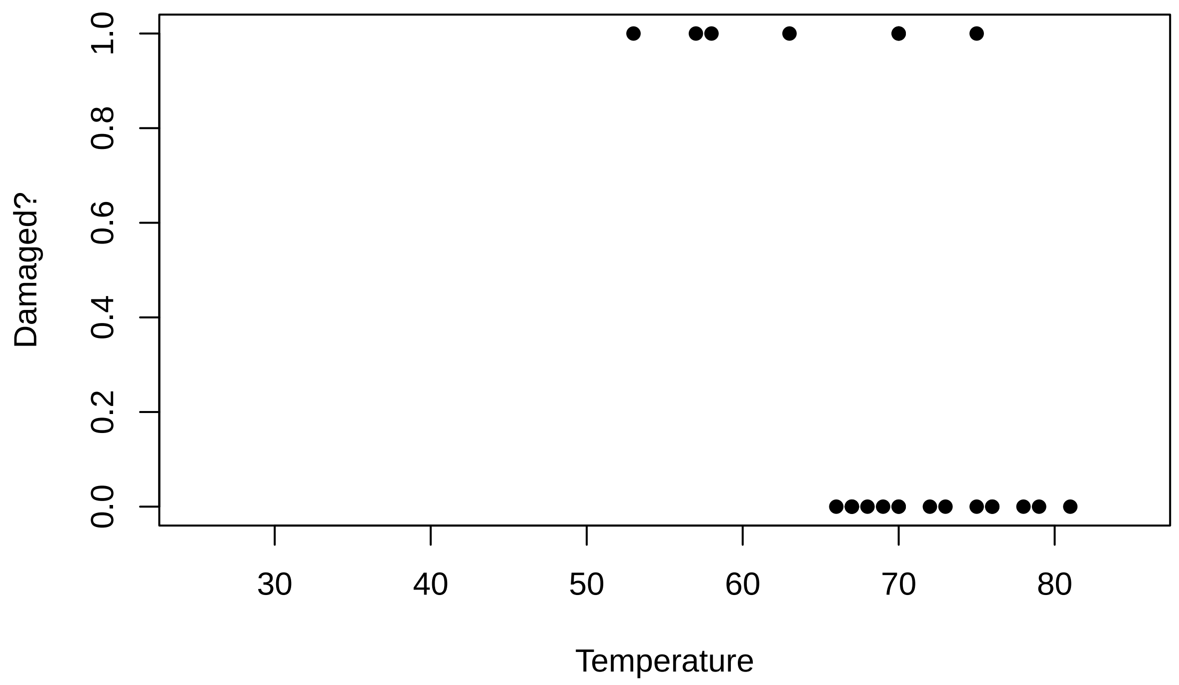
```
orings_damaged = orings[orings$damage,]
plot(damage ~ temp, orings_damaged, pch = 16, ylim = c(0,1),
     xlab="Temperature", ylab="Damaged?")
```



It seems plausible that the distribution of temperature is uniform. At least, NASA thought so, because they went ahead with the launch.

Now, however, we plot all the observations, including those when no damage was observed.

```r
plot(damage ~ temp, orings, pch = 16, xlim=c(25,85), ylim = c(0,1),
     xlab="Temperature", ylab="Damaged?")
```



You don't need to be a statistician to see that there's a problem.

The point here is that you should always make sure your analyses are pertinent to the question you are ultimately interested in answering. NASA was interested in predicting the probability of damage at a given launch temperature. But their EDA only gave information about the distribution of temperature given damage.

Logistic regression with O-rings data

We fit a logistic regression to estimate the probability of damage at a given temperature.

```
l.glm = glm(damage ~ temp, family=binomial, orings)
summary(l.glm)
```

```
##
## Call:
## glm(formula = damage ~ temp, family = binomial, data = orings)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  15.0429     7.3786   2.039   0.0415 *
## temp         -0.2322     0.1082  -2.145   0.0320 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 28.267  on 22  degrees of freedom
## Residual deviance: 20.315  on 21  degrees of freedom
## AIC: 24.315
##
## Number of Fisher Scoring iterations: 5
```

Interpretation: a decrease in temperature by one degree Fahrenheit is associated with an increase in the estimated odds of O-ring failure by a multiplicative factor of $e^{0.23} = 1.25$.

Logistic regression with O-rings data: Confidence intervals

First we calculate 95% confidence intervals by hand (Remember that $z^{(1-\alpha/2)} = 1.96$.)

```
beta.temp = l.glm$coef["temp"]
se.temp = summary(l.glm)$coef["temp","Std. Error"]
beta.temp.upper = beta.temp + 1.96*se.temp
beta.temp.lower = beta.temp - 1.96*se.temp
c(beta.temp.lower,beta.temp.upper)
```

```
##        temp        temp
## -0.44430615 -0.02001934
```

Logistic regression with O-rings data: Confidence intervals

The automatic intervals given by `confint` look a little different.

```
confint(l.glm)
```

```
## Waiting for profiling to be done...
```

```
##                  2.5 %      97.5 %
## (Intercept)  3.3305848 34.34215133
## temp        -0.5154718 -0.06082076
```

To explain the difference, remember that for the linear model we had two added variable tests – the T-test, and a special case of the partial F-test – which turned out to always give the same p-value. (You saw an example of this in Homework 3.) For GLMs we still have two added variable tests – an analog of the T-test, based on asymptotic Normality of parameters, and an analog of the partial F-test, based on the asymptotic χ^2 distribution of the log-likelihood – but they are only the same asymptotically. The confidence interval given by `confint` comes from inverting the second kind of test.

The intervals given by the "by hand" and "automatic" methods are usually pretty similar. And they are asymptotically equivalent. For small number of samples the intervals given by `confint` are slightly better.

Logistic regression with O-rings data: Prediction

To compute a predicted probability of failure at say `temp = 60` degrees, we can first compute the predicted log-odds, then transform to probability scale using the inverse of the logit function, i.e. the logistic function.

```r
# Logistic function
ilogit = function(eta) exp(eta)/(1 + exp(eta))

# Estimated log odds of failure
eta_60 = predict(l.glm, newdata= data.frame(temp=60),type = "link")
c("Estimate of P(Y = 1|Temp  = 60)" = as.numeric(ilogit(eta_60)))
```

```
## Estimate of P(Y = 1|Temp  = 60)
##                         0.7527135
```

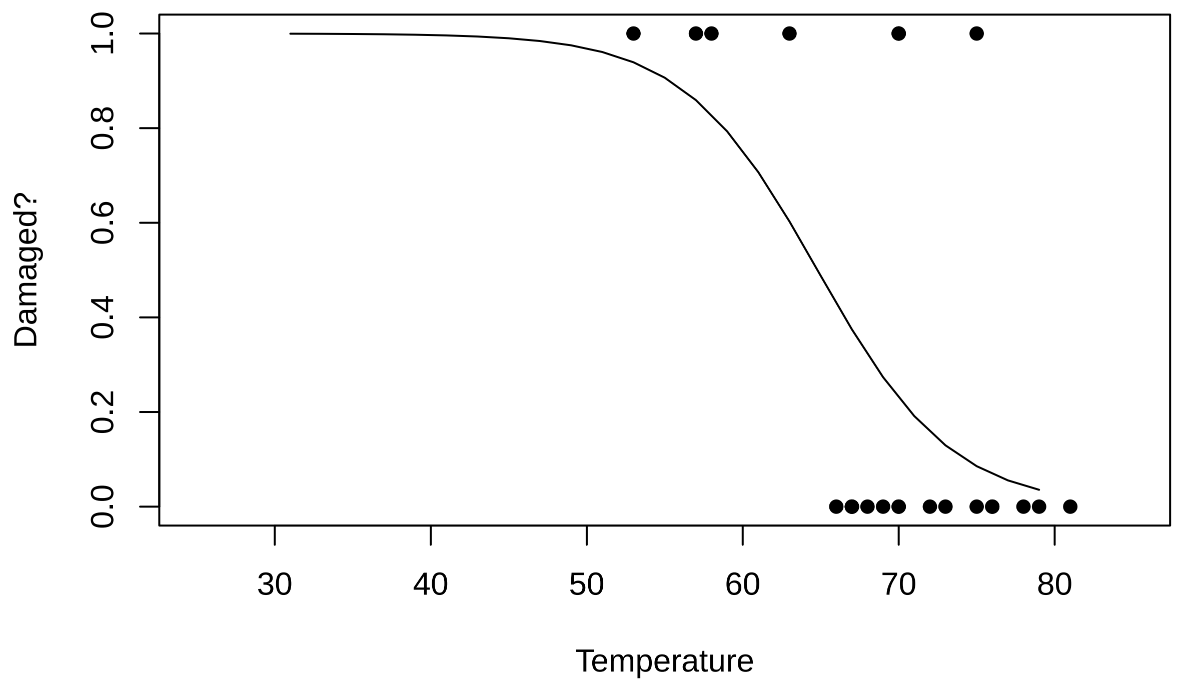
```r
# Note: this is the same as
#
# predict(l.glm, newdata= data.frame(temp=60),type = "response")
```

Logistic regression with O-rings data: Prediction

Finally, we compute and plot a predicted probability of failure for a range of temperatures.

```r
# Predicitions for probability of failure.
temps = seq(31,80, by=2)
m_pred = predict(l.glm, newdata= data.frame(temp=temps), type = "response")

# Plot
plot(damage ~ temp, orings, pch = 16, xlim=c(25,85), ylim = c(0,1),
     xlab="Temperature", ylab="Damaged?")
lines(temps,m_pred)
```



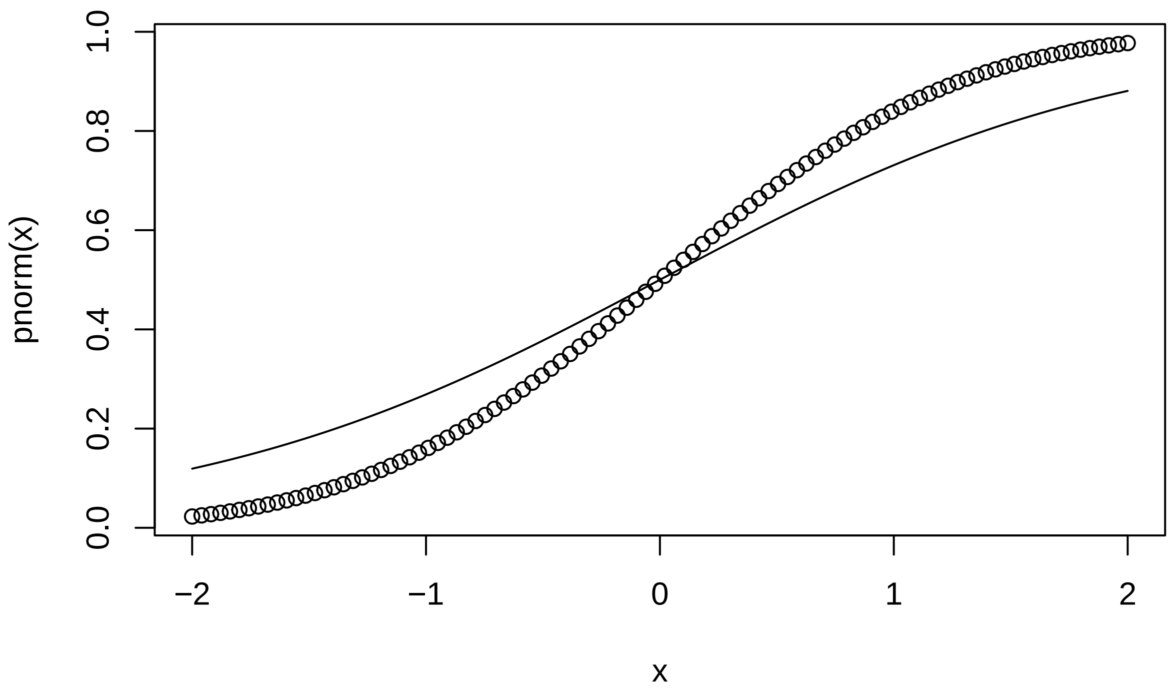
Based on our logistic regression, at a launch temperature of 31° we estimate that there was a 100% probability the O-rings would sustain damage!

Of course, our estimated probability is itself uncertain, and so will have some error even if the logistic model is correct. This means we should probably attach a confidence interval to it. We will discuss this more next time…

Link functions

Let's look at the inverse logit function and the inverse probit functions.

```r
x  = seq(-2,2,length.out = 100)
plot(x,pnorm(x))
lines(x,exp(x)/(1 + exp(x)))
```



Both functions are fairly similar. We also see that both allow for relatively sharp "phase transitions" where $m(x)$ can go from close to 0 to close to 1 quite quickly.