

# Numerical Optimization

*Function optimization* refers to the problem of finding a value of  $x$  to make some function  $f(x)$  as large (or as small) as possible.

In statistics, these problems often arise in the context of calculating estimates of model parameters.

- Nonlinear least squares
- Generalized linear models
- Maximum likelihood estimates

Sometimes we can find the solution explicitly, for example using the derivatives of  $f$ . But when the solution can't be found in closed form, we turn to *numerical optimization*.

There are very many techniques for numerical optimization, and we can't possibly cover them all.

We'll talk about two basic methods and how to program them:

- Golden section search
- Newton-Raphson algorithm

Then we'll move on to some statistical examples and how to use the built-in optimization methods in R.

Since every maximization problem can be rewritten as a minimization problem, using  $-f(x)$  rather than  $f(x)$ , we'll assume from now on that we're minimizing.

## Golden section search

Assume  $f(x)$  has a *single minimum* on the interval  $[a,b]$ .

The *golden section search* algorithm iteratively shrinks the interval over which we're looking for the minimum, until the length of the interval is less than some preset tolerance.

The name golden section comes from the fact that at each iteration, we choose a new point to evaluate so that we can reuse one of the points from the last iteration. It works out that the way to do this is to maintain the so-called *golden ratio* between the distances between points.

Consider two line segments  $c$  and  $d$ . They are said to be in golden ratio if their sum  $c+d$  is to  $c$  as  $c$  is to  $d$ .

$$\begin{aligned}\frac{c+d}{c} &= \frac{c}{d} = \phi \\ \Rightarrow \frac{d\phi + d}{d\phi} &= \frac{d\phi}{d} \\ \Rightarrow \frac{\phi + 1}{\phi} &= \phi \\ \Rightarrow \phi^2 - \phi - 1 &= 0 \\ \Rightarrow \phi &= \frac{1 + \sqrt{5}}{2} \approx 1.618034\end{aligned}$$



An example:

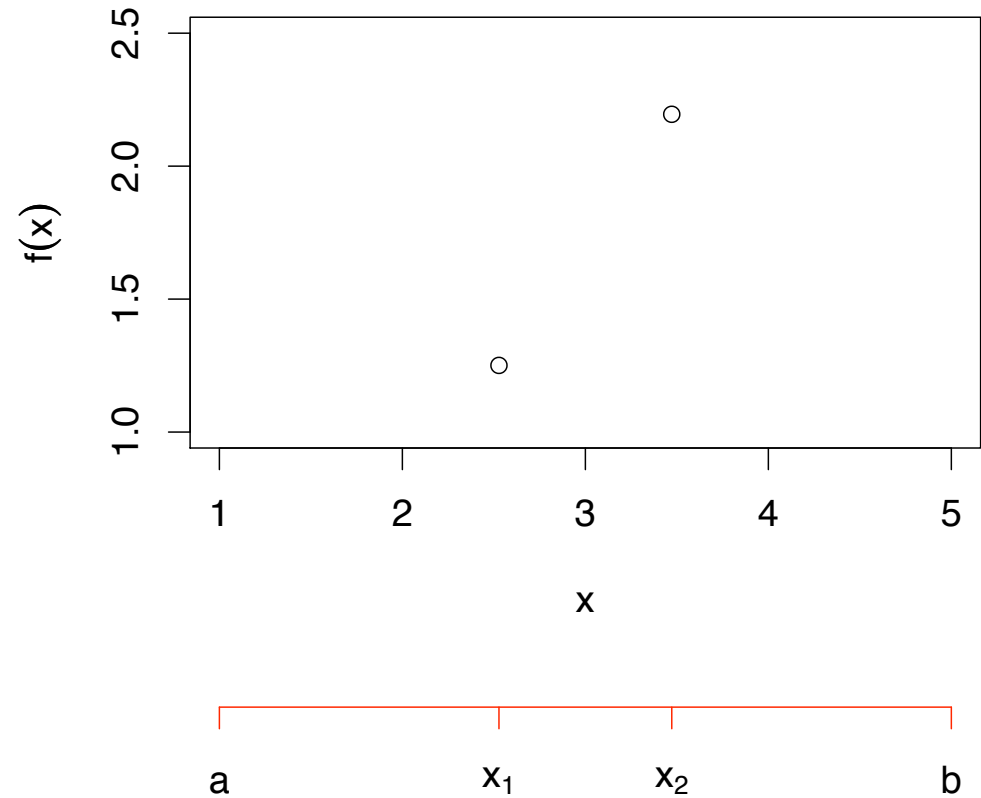
Start with

$$x_1 = b - (b - a)/\phi$$

$$x_2 = a + (b - a)/\phi$$

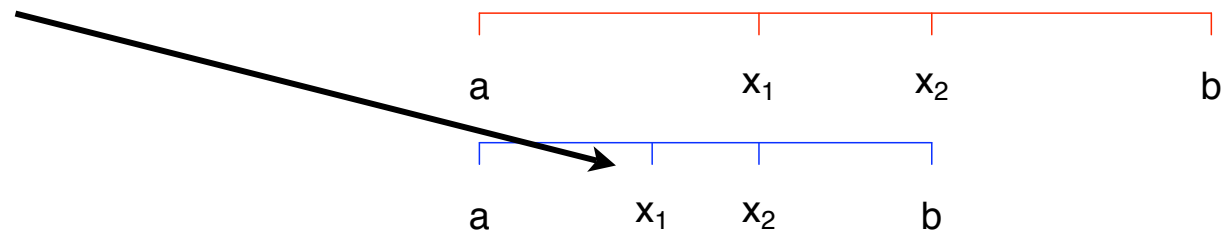
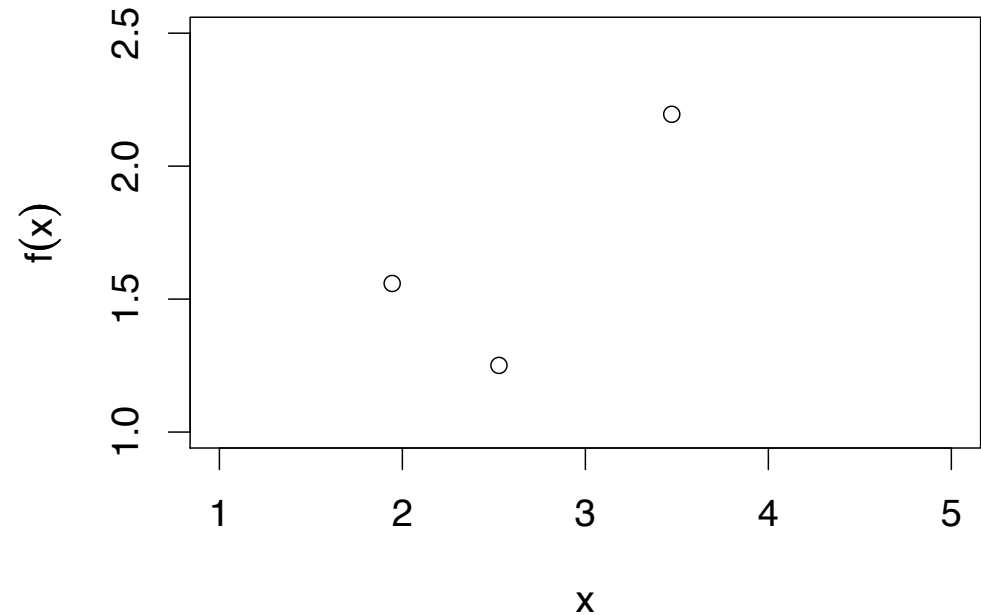
Now compare  
 $f(x_1)$  and  $f(x_2)$ .

Since  $f(x_1) < f(x_2)$ ,  
we know the minimum  
must be in  $[a, x_2]$ .



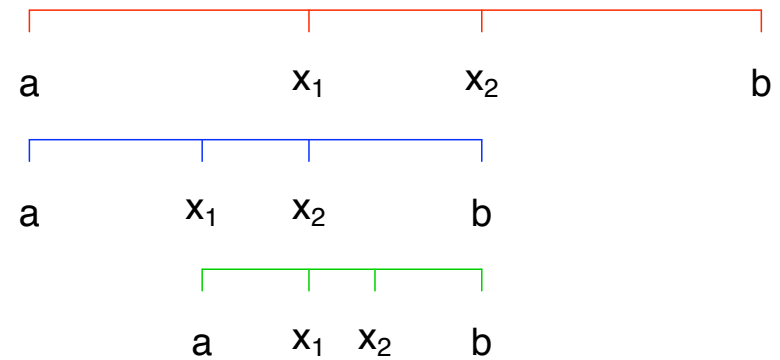
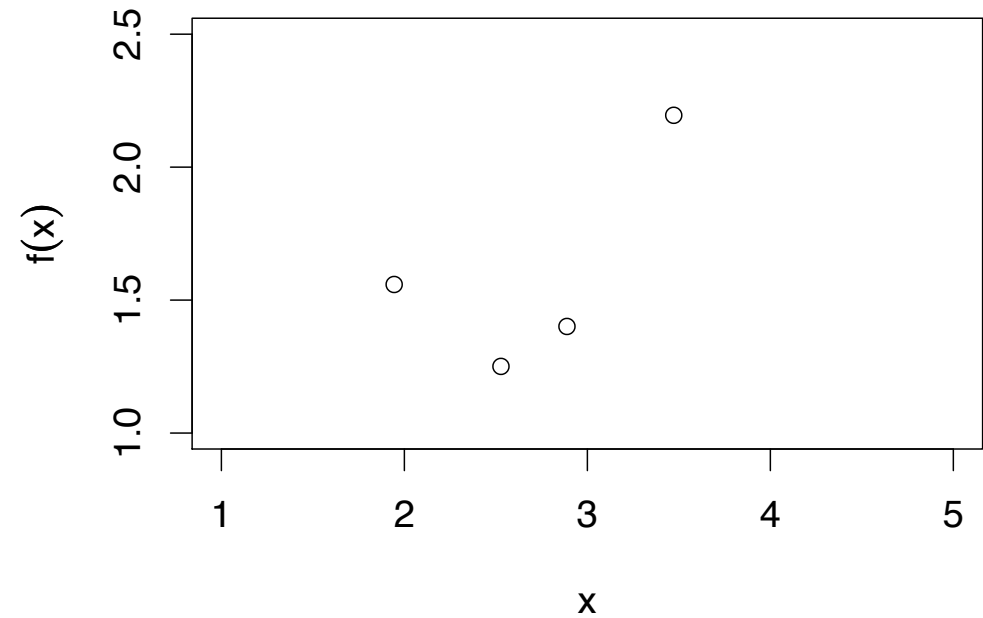
## An example

Add a new point  
and maintain the  
golden ratio.



This time  $f(x_1) > f(x_2)$ , so the  
minimum must be in  $[x_1, b]$ .

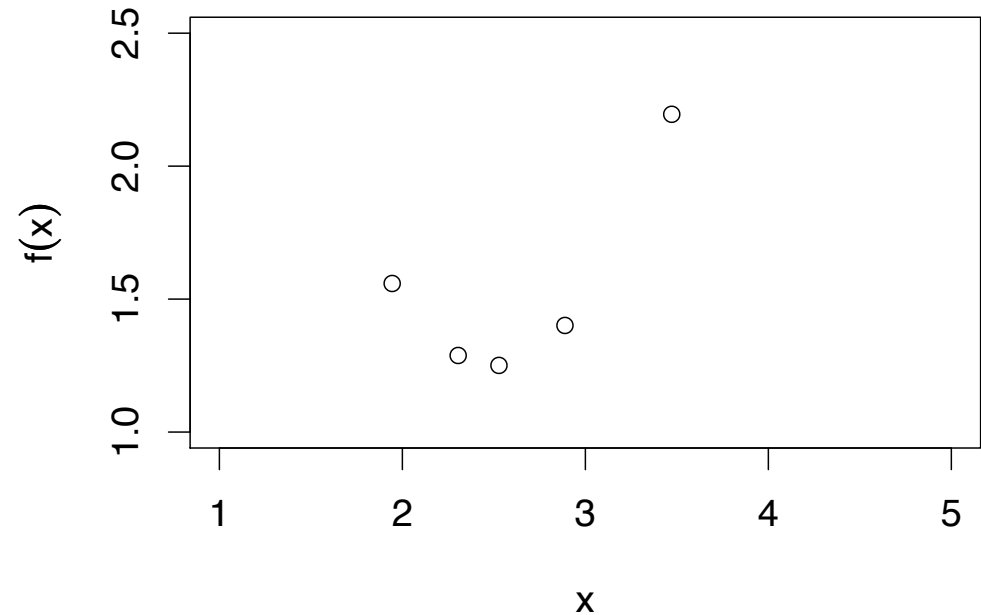
## An example



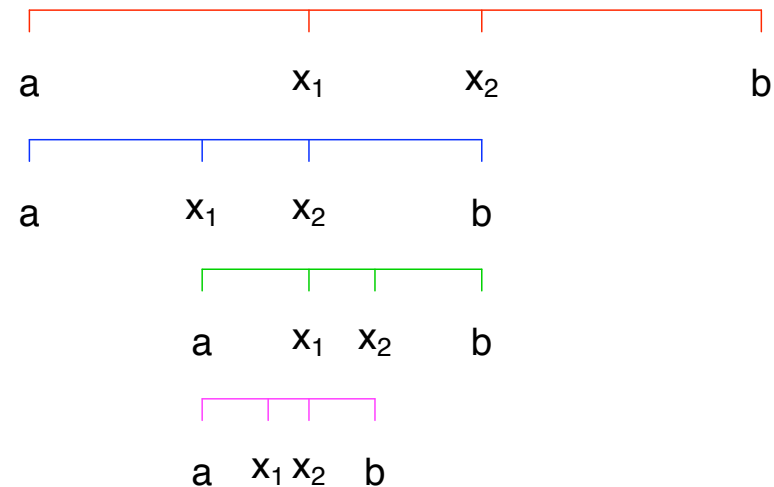
Keep going like this....



## An example



When  $b-a$  is sufficiently small, we stop and report a minimum of  $(a+b)/2$ . One can show that the error is at most  $(1-\Phi)(b-a)$ .



The *Newton-Raphson algorithm* can be used if the function to be minimized has two continuous derivatives that may be evaluated.

Again assume that there is a single minimum in  $[a,b]$ . If the minimizing value  $x^*$  is not at  $a$  or  $b$ , then  $f'(x^*) = 0$ . If in addition  $f''(x^*) > 0$ , then  $x^*$  is a minimum.

The main idea behind N-R is that if we have an initial value  $x_0$  that is close to the minimizing value, then we can approximate

$$f'(x) \approx f'(x_0) + (x - x_0)f''(x_0)$$

$$f'(x) \approx f'(x_0) + (x - x_0)f''(x_0)$$

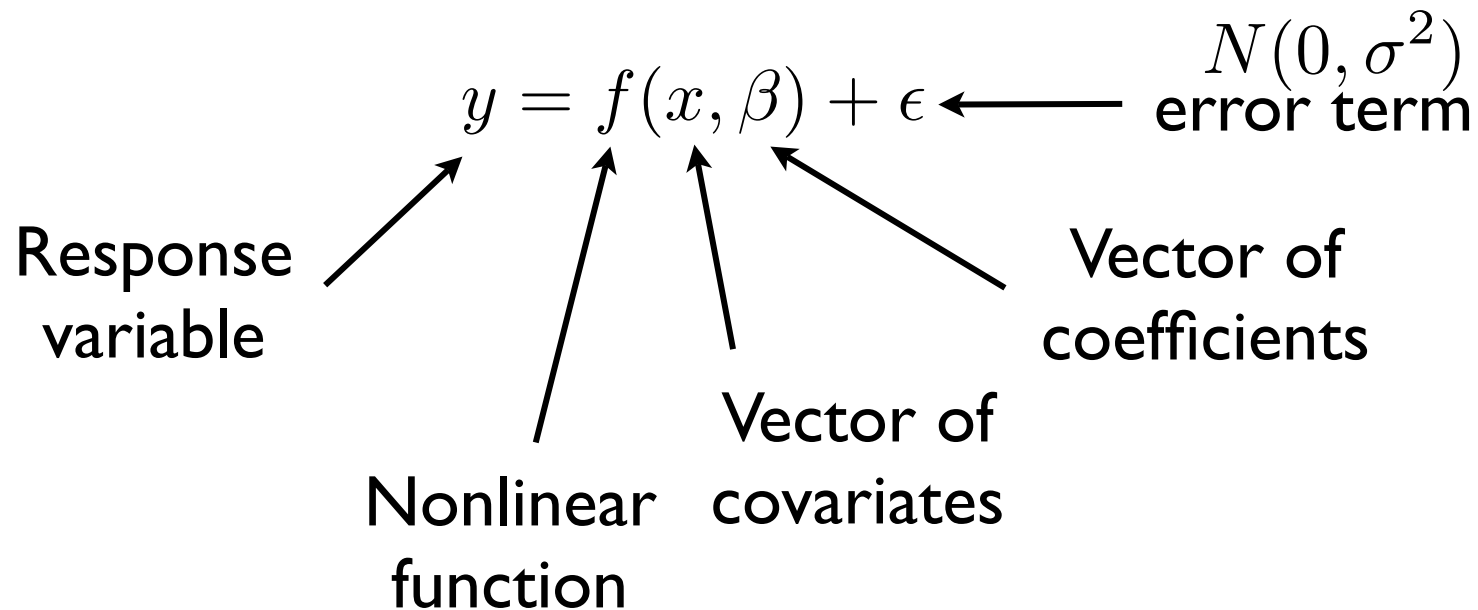
Setting RHS = 0 gives  $x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)}$

We keep going in this way until  $f'(x_n)$  is sufficiently close to zero.

It's important to have a good initial guess, otherwise the Taylor series approximation may be very poor and we may even have  $f(x_{n+1}) > f(x_n)$ .

We've already seen one example of numerical optimization in action, when we used nonlinear least squares to fit a curve to the covariogram for spatial data.

This is useful more generally, for non-linear regression models of the form

$$y = f(x, \beta) + \epsilon$$


Response variable

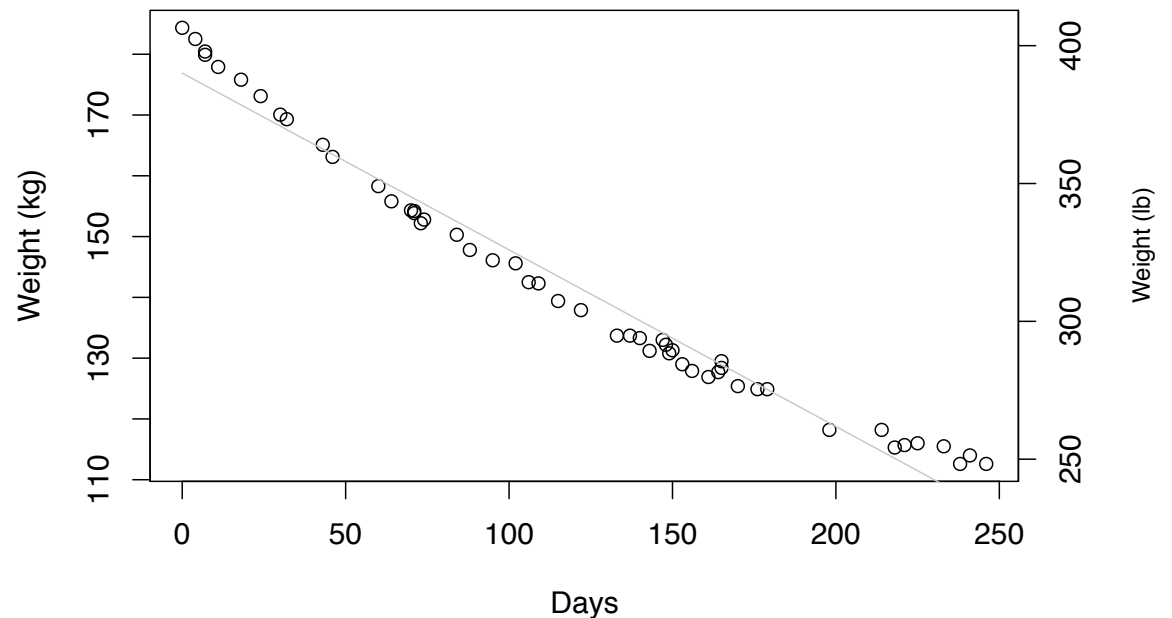
Nonlinear function

Vector of covariates

Vector of coefficients

$N(0, \sigma^2)$   
error term

## Example: weight loss



Patients tend to lose weight at diminishing rate. Here is data from one patient with a linear fit superimposed.

Another proposed model is  $y = \beta_0 + \beta_1 2^{-t/\theta} + \epsilon$

Ultimate lean  
weight (asymptote)

Total amount  
to be lost

Time taken to  
lose half amount  
remaining to be lost

The function `nls` in R uses numerical optimization to find the values of the parameters that minimize the sum of squared errors

$$\sum_{i=1}^n (Y_i - f(x_i, \beta))^2$$

The main arguments are

- `formula` - outcome on the LHS, function on the RHS
- `data` - dataframe holding the variables
- `start` - vector of starting values

# Numerical Optimization II: Fitting Generalized Linear Models

The normal linear model assumes that

- 1) the expected value of the outcome variable can be expressed as a *linear* function of the explanatory variables, and
- 2) the residuals (observations minus their expected values) are independent and identically distributed with a *normal* distribution.

Last time we talked about relaxing assumption 1), using nonlinear regression models.

Today we'll talk about relaxing assumption 2), using what are called *generalized linear models*.



First, a few words about the normal linear model.

With a single explanatory variable, it has the form

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

where  $\epsilon_i \stackrel{iid}{\sim} N(0, \sigma^2)$ ,  $i = 1, \dots, n$

Recall that the least squares estimates of  $\beta_0$  and  $\beta_1$  minimize the *residual sum of squares*

$$RSS(\beta_0, \beta_1) = \sum_{i=1}^n (Y_i - \beta_0 - \beta_1 X_i)^2$$

With a little calculus, we can minimize RSS explicitly....

$$\partial RSS / \partial \beta_0 = -2 \sum_{i=1}^n (Y_i - \beta_0 - \beta_1 X_i)$$

$$\partial RSS / \partial \beta_1 = -2 \sum_{i=1}^n (Y_i - \beta_0 - \beta_1 X_i) X_i$$

Setting each equal to zero and solving, we get

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2}$$

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}$$

In this model, the least squares estimates are equal to the maximum likelihood estimates, which we'll discuss next time.

We can do similar calculations if we have more than one explanatory variable.

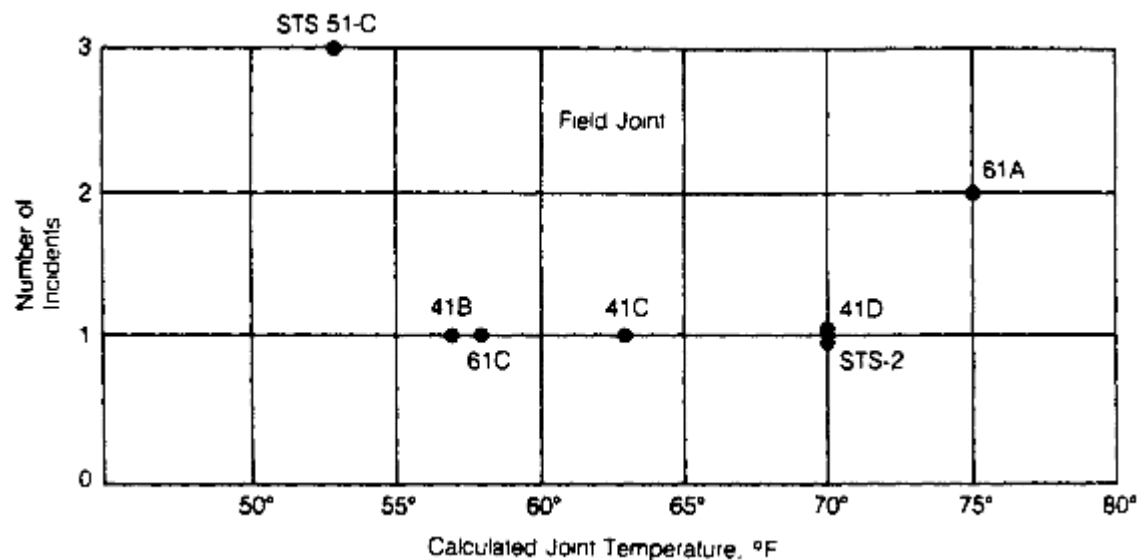
Another way of thinking about what we've done in the normal linear model is that we've expressed the mean of the Y's as a linear combination of the X's.

$$\begin{aligned} E[Y_i] &= \beta_0 + \beta_1 X_i + E[\epsilon_i] \\ &= \beta_0 + \beta_1 X_i \end{aligned}$$

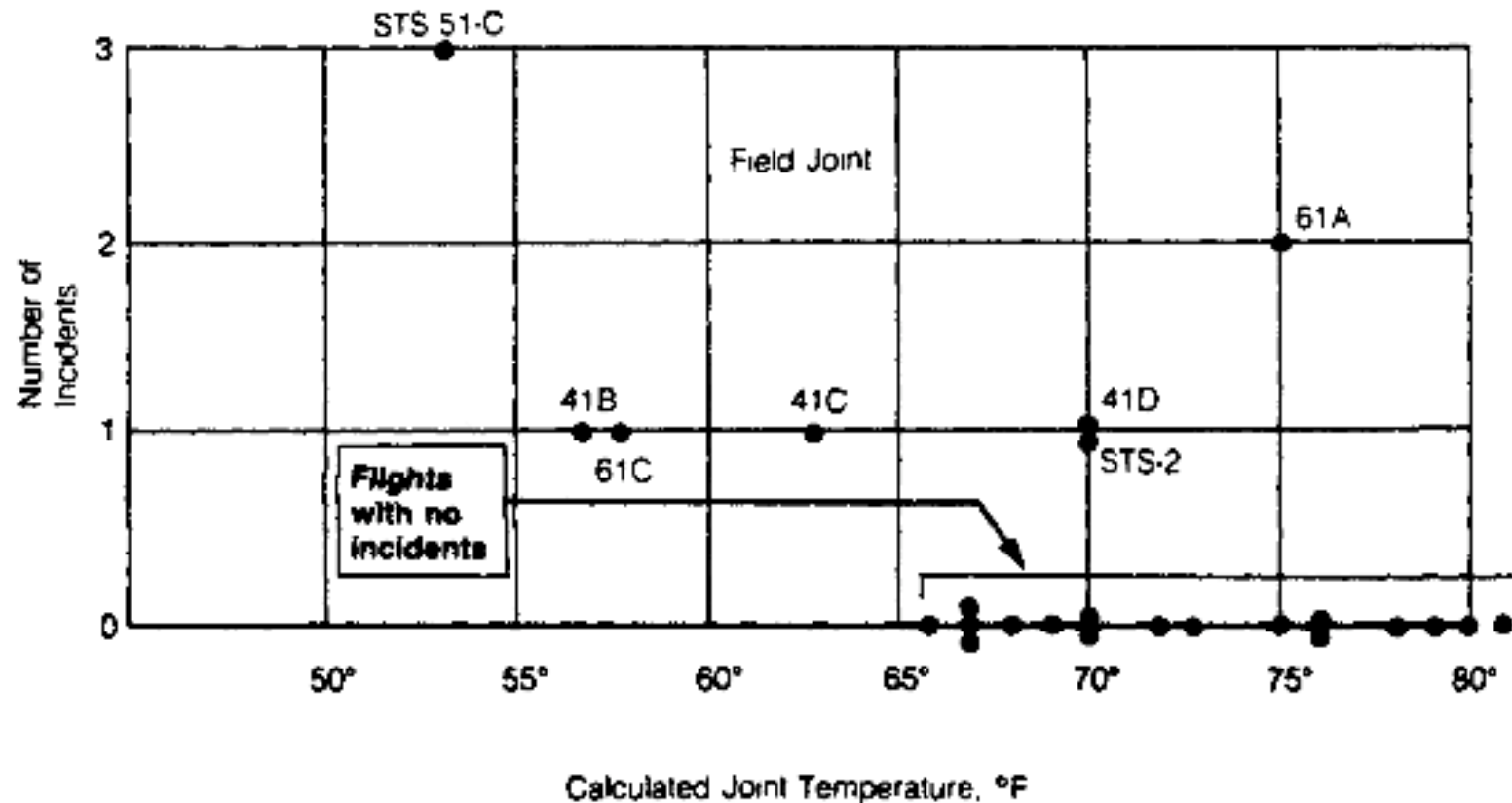
To work with non-normal distributions, we're going to slightly modify this idea.

First, a motivating example.

Prior to the launch of the space shuttle Challenger, there was some debate about whether temperature had any effect on the performance of a key part called an O-ring. The following plot, with data from past flights, was used as evidence that it was safe to launch at a temperature of  $31^{\circ}\text{F}$ .

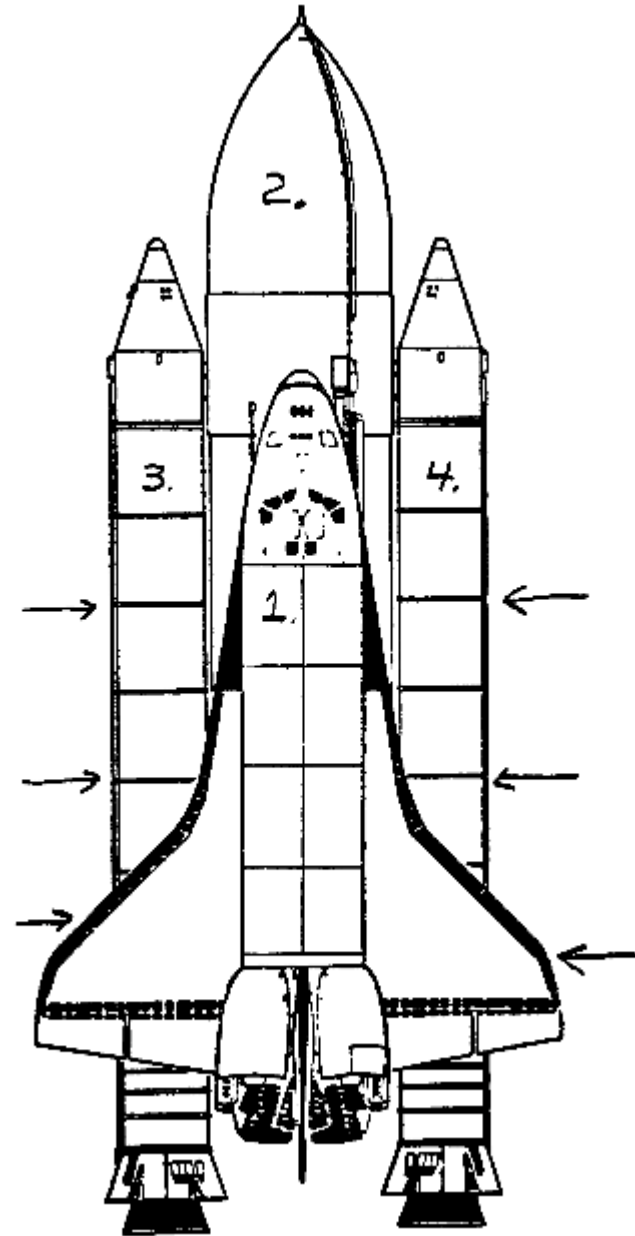


One key problem with this analysis was that the engineers left out the data from all the flights with no O-ring problems, under the mistaken assumption that these gave no extra information.



The solid rocket motors (labeled 3 and 4) are delivered to Kennedy Space Center in four pieces, and they are connected on site using the O-rings. There are actually two sets of O-rings at each joint, but we'll focus on the primary ones.

So in each launch, there are six primary O-rings that can fail. If any one fails, it can lead to a catastrophic failure of the whole shuttle.



Here are the data on past failures of the primary O-rings.

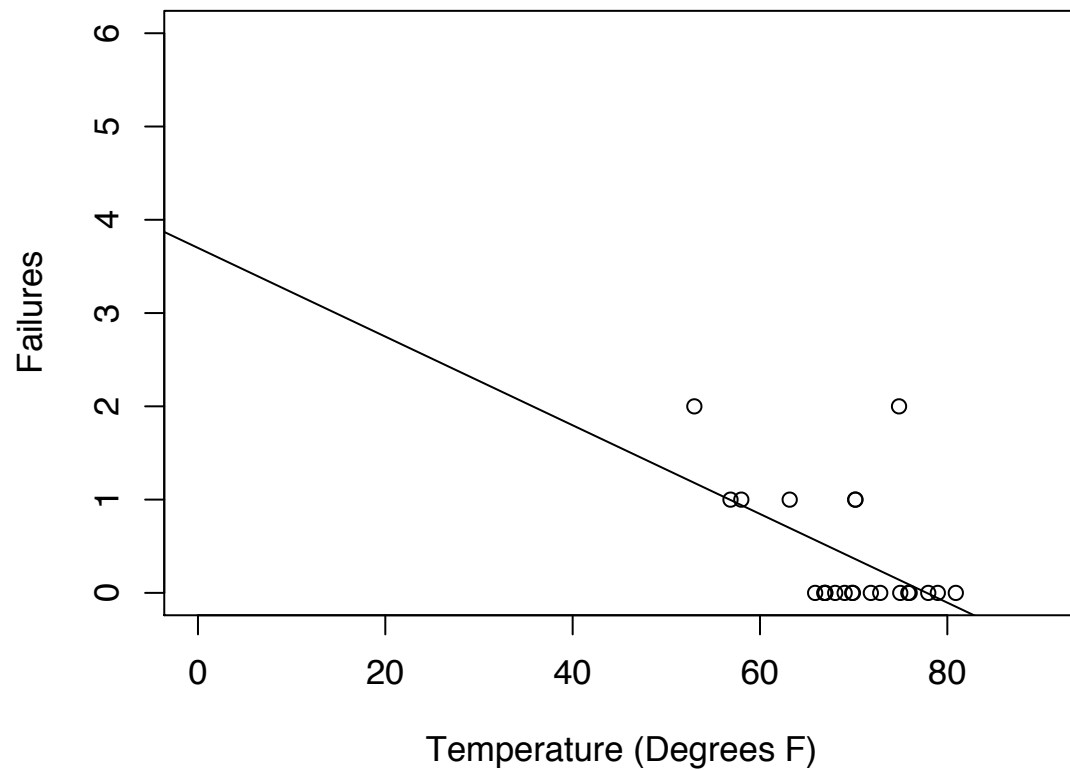
The data from past flights come from rocket motors that are retrieved from the ocean after the flight. There had been 24 shuttle launches prior to Challenger, of which the rocket motors were retrieved in 23 cases.

	Temp	Fail	Date
1	66	0	4/12/81
2	70	1	11/12/81
3	69	0	3/22/82
4	68	0	11/11/82
5	67	0	4/4/83
6	72	0	6/18/83
7	73	0	8/30/83
8	70	0	11/28/83
9	57	1	2/3/84
10	63	1	4/6/84
11	70	1	8/30/84
12	78	0	10/5/84
13	67	0	11/8/84
14	53	2	1/24/85
15	67	0	4/12/85
16	75	0	4/29/85
17	70	0	6/17/85
18	81	0	7/29/85
19	76	0	8/27/85
20	79	0	10/3/85
21	75	2	10/30/85
22	76	0	11/26/85
23	58	1	1/12/86

We could fit a linear regression model to this data, relating the expected number of failures to temperature.

Some problems with this approach are that

- 1) the residuals are clearly not iid normal
- 2) if we go out far enough, we actually predict a negative number of failures.



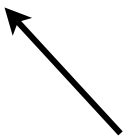


Instead we will fit a *logistic regression model*.

This model is appropriate when the data have a binomial distribution (counting the number of events out of  $n$  trials), of which binary data is a special case with  $n=1$ .

The expected value for a given trial is  $p_i$ , the probability of an event when the explanatory variable  $X = X_i$ . We relate this to the linear predictor using the *logit function*.

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



This ratio is called, the odds, so the logit can also be called the log odds.

The case we are discussing (binomial outcome, logit function) is a special case of a larger class of models called *generalized linear models*.

Some other examples:

Normal outcome, identity link

$$Y_i \sim N(\mu_i, \sigma^2),$$

$$\mu_i = \beta_0 + \beta_1 X_i$$

Poisson outcome, log link

$$Y_i \sim Pois(\lambda_i)$$

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

Note that in each case, the link function maps the space of the parameter representing the mean of the distribution ( $\mu_i$ ,  $\lambda_i$ , or  $p_i$ ) to the real line, which is the space of the linear predictor.

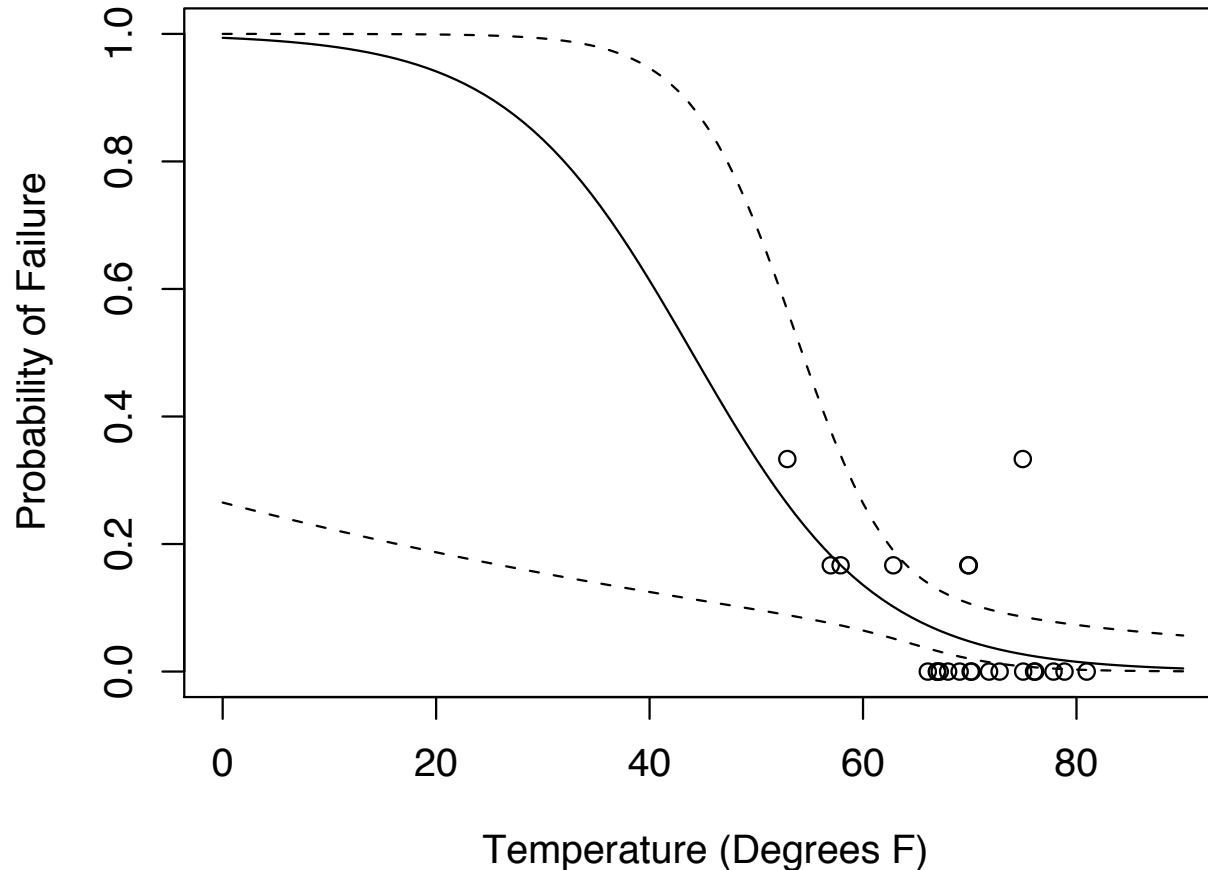
Generalized linear models can be fit using an algorithm called *iteratively reweighted least squares*. In R, this is implemented in the function `glm`.

```
# First create a matrix with events and non-events
FN <- cbind(challenge$Fail, 6 - challenge$Fail)
```

```
# Fit using specified family, default link function
glm.fit <- glm(FN~Temp, data = challenge,
               family = binomial)
```

```
# Now predict for a range of temperatures
tempseq <- seq(0, 90, length = 100)
pred <- predict(glm.fit, newdata = data.frame(Temp =
tempseq), se.fit = TRUE)
inv.logit <- function(x){1/(1+exp(-x))}
lines(tempseq, inv.logit(pred$fit))
lines(tempseq, inv.logit(pred$fit + 2*pred$se.fit), lty = 2)
lines(tempseq, inv.logit(pred$fit - 2*pred$se.fit), lty = 2)
```

The confidence interval at 31°F is quite wide, but the point estimate probably still should have been cause for alarm, especially since the temperature was colder than anything that had been tried before.



Interpretation of the coefficients is a bit trickier in logistic regression models than it is in linear regression models.

When  $E[Y_i] = \beta_0 + \beta_1 X_i$ , we can say that

- $\beta_0$  is the expected value of  $Y$  when  $X_i = 0$   
(This is not always interesting; for example the temperature will almost never be  $0^\circ\text{F}$ .)
- $\beta_1$  is the change in the expected value of  $Y$  due to a unit increase in  $X$ .

Now we have  $\log \left( \frac{p_i}{1 - p_i} \right) = \beta_0 + \beta_1 X_i$

We can interpret the parameters in terms of log odds, odds, or probabilities.

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

The interpretation regarding log odds is the easiest to state but probably the hardest to understand.

- $\beta_0$  is the log odds of an event when  $X_i = 0$
- $\beta_1$  is the change in the log odds due to a unit increase in  $X$ .

Now, if we exponentiate both sides, we get

$$p_i / (1 - p_i) = \exp(\beta_0 + \beta_1 X_i)$$

which implies that  $\exp(\beta_0)$  is the value of the odds when  $X_i = 0$

Also, suppose  $X_i = X_j + 1$ .

$$\begin{aligned}\frac{p_i/(1 - p_i)}{p_j/(1 - p_j)} &= \frac{\exp(\beta_0) \exp(\beta_1 X_i)}{\exp(\beta_0) \exp(\beta_1 X_j)} \\ &= \exp(\beta_1 (X_i - X_j)) = \exp(\beta_1)\end{aligned}$$

So  $\exp(\beta_1)$  gives the *multiplicative* change in odds corresponding to a one unit change in  $X$ .

In particular, if  $X$  takes only the values 0 and 1, then  $\exp(\beta_1)$  is the *odds ratio* for category 1 compared to category 0.

The interpretation in terms of probabilities is conditional on other variables in the model, so we'll save it for after we talk about using multiple regressors.

## Another example, this time with multiple explanatory variables

```
> library(MASS)
> birthwt[1:2,]
      low age lwt race smoke ptl ht ui ftv  bwt
85     0  19 182    2     0   0  0  1   0 2523
86     0  33 155    3     0   0  0  0   3 2551
```

'low' indicator of birth weight less than 2.5kg

'age' mother's age in years

'lwt' mother's weight in pounds at last menstrual period

'race' mother's race ('1' = white, '2' = black, '3' = other)

'smoke' smoking status during pregnancy

'ptl' number of previous premature labours

'ht' history of hypertension

'ui' presence of uterine irritability

'ftv' number of physician visits during the first trimester

'bwt' birth weight in grams



We are now confronted with the question of model choice. There are a variety of principles that can guide us here, but in the interest of time, let's consider one criterion balancing goodness of fit with parsimony (the number of parameters).

The Akaike information criterion is

$$AIC = 2k - 2 \log(L)$$

where  $k$  is the number of parameters in the given model and  $L$  is the maximized value of the likelihood for that model. (For now, you can think of the likelihood as the joint density of the data for a particular setting of the parameter values.) Looking at this criterion, we favor models with a *lower* value of AIC.

# Numerical Optimization II: Maximizing Likelihoods

One of the canonical cases in which we need to numerically optimize a function in statistics is to find the *maximum likelihood estimate*.

For  $X_1, \dots, X_n \stackrel{iid}{\sim} f(x; \theta)$ , the *likelihood function* is

$$L(\theta) = \prod_{i=1}^n f(X_i; \theta)$$

The *log-likelihood function* is

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^n \log f(X_i; \theta)$$

The *maximum likelihood estimator* (MLE), which we'll denote by  $\hat{\theta}$ , is the value of  $\theta$  that maximizes  $L(\theta)$ . (Note that this is equivalent to maximizing  $\ell(\theta)$ ).

Another important thing to note is that we can multiply the likelihood by a constant (or add a constant to the log-likelihood), and this does not change the location of the maximum.

Therefore, we often work only with the part of the likelihood that concerns  $\theta$ . This part of the function is called the *kernel*.

In simple cases, we can often find the MLE in closed form by, for example, differentiating the log-likelihood with respect to  $\theta$ , setting this equal to zero, and solving for  $\theta$ . But things are often not this simple!

As an example, let's go back to the logistic regression model. Remember, we have  $Y_i \sim \text{Ber}(p_i)$ ,  $i = 1, \dots, n$  where, inverting the logit function, we have

$$p_i = \frac{\exp\{\beta_0 + \beta_1 X_i\}}{1 + \exp\{\beta_0 + \beta_1 X_i\}}$$

and the likelihood function is  $L(\beta_0, \beta_1) = \prod_{i=1}^n p_i^{Y_i} (1 - p_i)^{1 - Y_i}$ , substituting in the expression above.

We can't maximize this analytically as a function of  $\beta_0$  and  $\beta_1$ , but we can easily write a function for the likelihood or log-likelihood and have R do the work for us....

```
# Function for the negative log-likelihood
```

```
logistic.nll <- function(beta, x, y, verbose = FALSE){  
  if(verbose) print(beta)  
  beta0 <- beta[1]; beta1 <- beta[2]  
  pvec <- exp(beta0 + beta1 * x) /  
    (1 + exp(beta0 + beta1 * x))  
  fvec <- y * log(pvec) + (1-y) * log(1 - pvec)  
  return(-sum(fvec))  
}
```

```
# Use optim to minimize the nll  
# par is a vector of starting values  
# better starting values => faster convergence, and  
# less chance of missing the global maximum
```

```
optim(par = c(0, 0), fn = logistic.nll,  
      x = x, y = y, verbose = TRUE)
```

In the case of logistic regression, minimizing the negative log-likelihood using `optim` will give the same answer as using `glm` with `family = binomial`.

However, there are many other models without built-in functions like `glm`. One example is the spatial models we discussed a few weeks ago.

Suppose we have a spatial field with mean zero and covariance function

$$Cov(Z(s_i), Z(s_j)) = \sigma^2 \exp\{-||s_i - s_j||/\rho\}$$

Before, we estimated  $\sigma^2$  and  $\rho$  by finding the covariogram and fitting a curve to it using nonlinear least squares.

However, the MLEs are actually much better estimators. The kernel of the likelihood function (for normal data) looks like this:

$$|\Sigma(\sigma^2, \rho)|^{-1/2} \exp\{-Z' \Sigma(\sigma^2, \rho)^{-1} Z/2\}$$

where  $Z = (Z_1, Z_2, \dots, Z_n)'$  is the vector of observations and  $\Sigma(\sigma^2, \rho)$  is the  $n$  by  $n$  matrix with

$$\begin{aligned} \Sigma(\sigma^2, \rho)_{i,j} &= \text{Cov}(Z(s_i), Z(s_j)) \\ &= \sigma^2 \exp\{-\|s_i - s_j\|/\rho\} \end{aligned}$$

We can again use `optim` to find the MLEs numerically....



A few words before we move on:

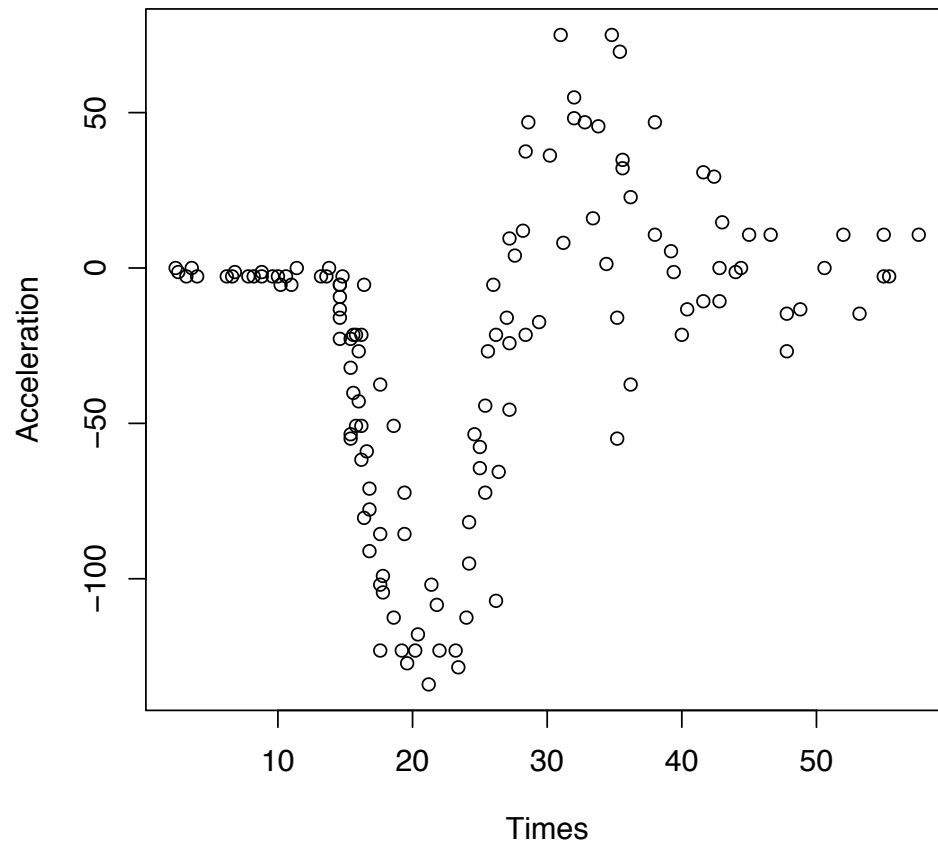
1) It's always preferable to find the MLEs in closed form if you can. The answer is exact, and you avoid all the errors that can be introduced in the numerical optimization, including possibly converging to a local rather than a global optimum.

2) If you do need to use numerical optimization, it's a good idea to evaluate the likelihood (or log-likelihood) over a grid of values first, to help you find a good starting value.

3) There's a lot more theoretical detail concerning MLEs that we don't have time to cover, importantly how to estimate uncertainty. See Stat 135.

# Nonparametric regression and scatterplot smoothing

We've looked at linear models and nonlinear models with a specified form, but what if you don't know a good function to relate two variables  $X$  and  $Y$ ?



This data set shows head acceleration in a simulated motorcycle accident, used to test helmets.

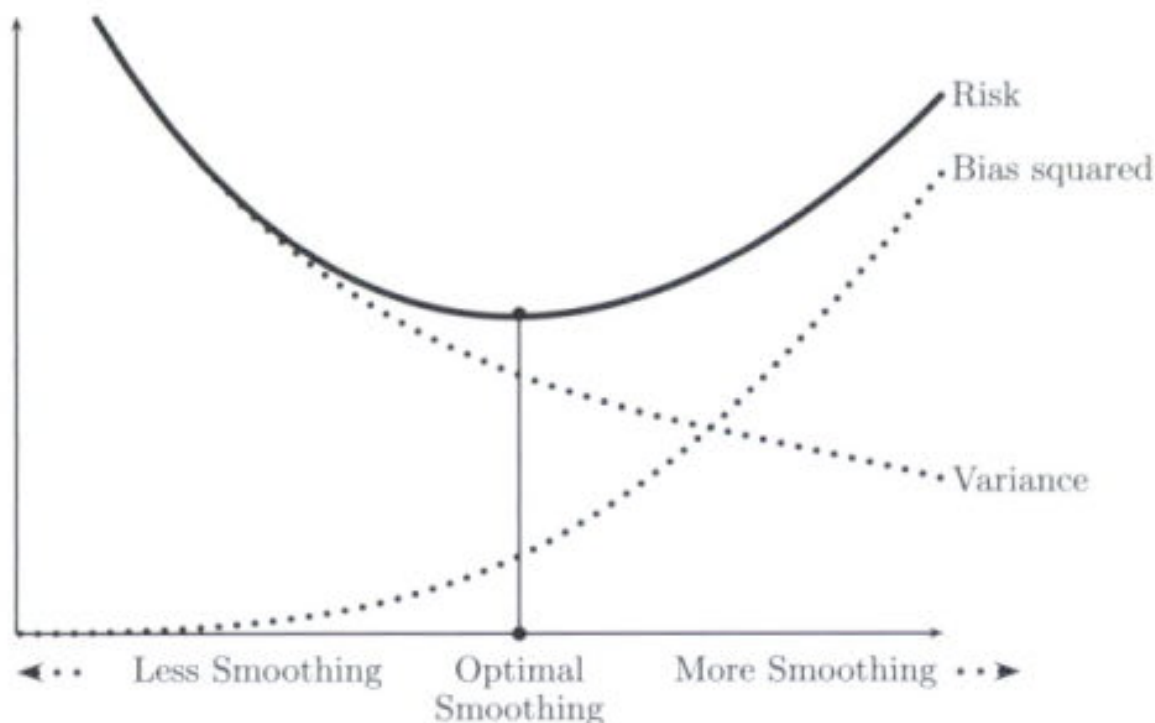
This area of statistics is known as *nonparametric regression* or *scatterplot smoothing*. Basically, we want to draw a curve through the data that relates  $X$  and  $Y$ . More formally, we suppose

$$Y_i = f(X_i) + \epsilon_i$$

where  $f$  is an unknown function and the  $\epsilon_i$  are iid with some common distribution, typically normal.

Now, if we don't put any restrictions on  $f$ , it's easy to get a perfect fit to the data -- just draw a curve that passes through all the points! But this curve is unlikely to give good predictions for any future observations.

Aside: This actually gets at a fundamental idea in statistics, called the *bias-variance tradeoff*. We can get a very low variance estimator of  $f$  by interpolating the data, using a very wiggly curve. But this introduces a lot of bias. So we look for a happy medium.



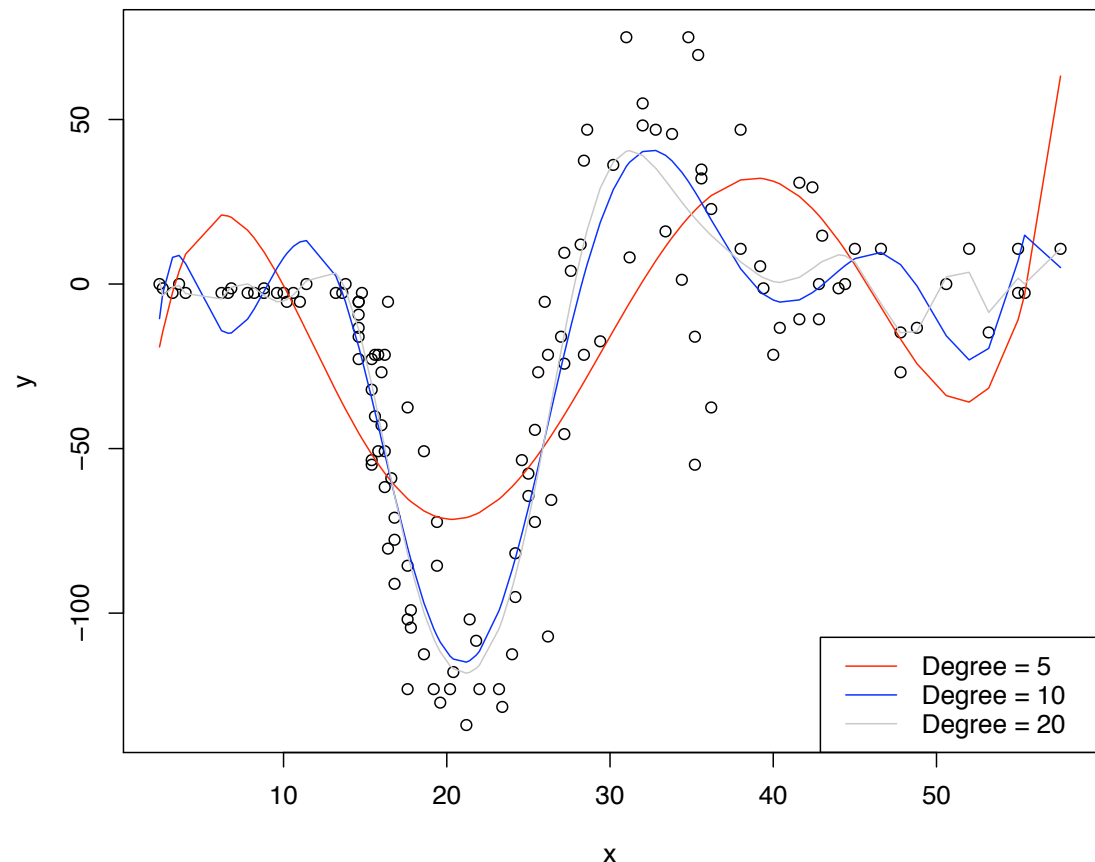
We won't cover the theoretical details, here, but just keep in mind this question of how much smoothing to do.

Back to the motorcycle data....

One of the simplest things we could do would be to fit a high degree polynomial.

But fitting a global polynomial this way isn't very efficient.

How about breaking up the region of  $x$  and fit a separate, lower-degree polynomial in each region?



This type of model is known as a *piecewise polynomial model* or *regression splines*.

The breakpoints, between which we have separate polynomial functions, are called *knots*.

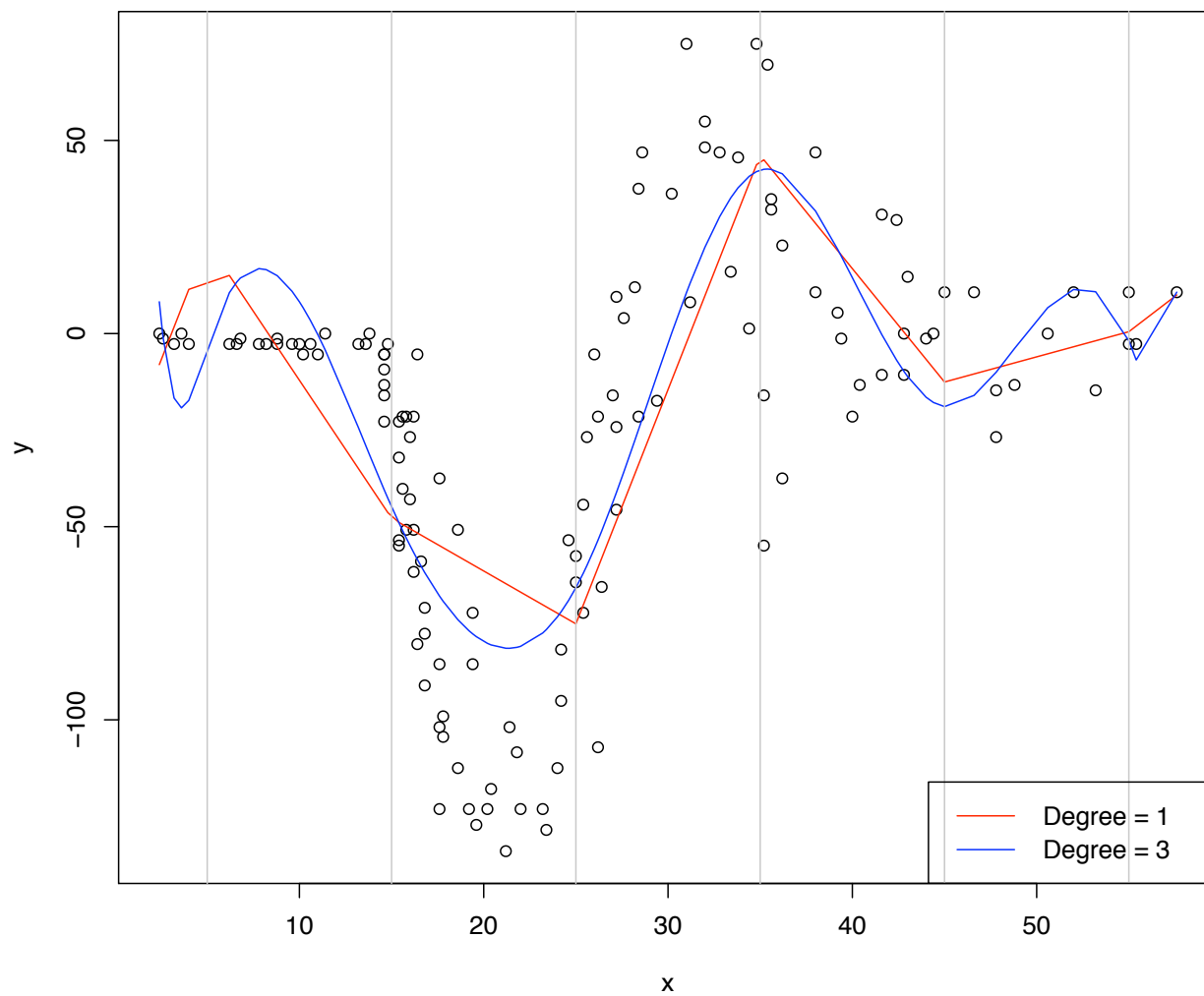
Typically we impose some constraints on the way the functions match up at the knots, such as maintaining the first and second derivatives.

So the modelling choices boil down to

- 1) where to put the knots
- 2) what degree polynomial to fit between the knots

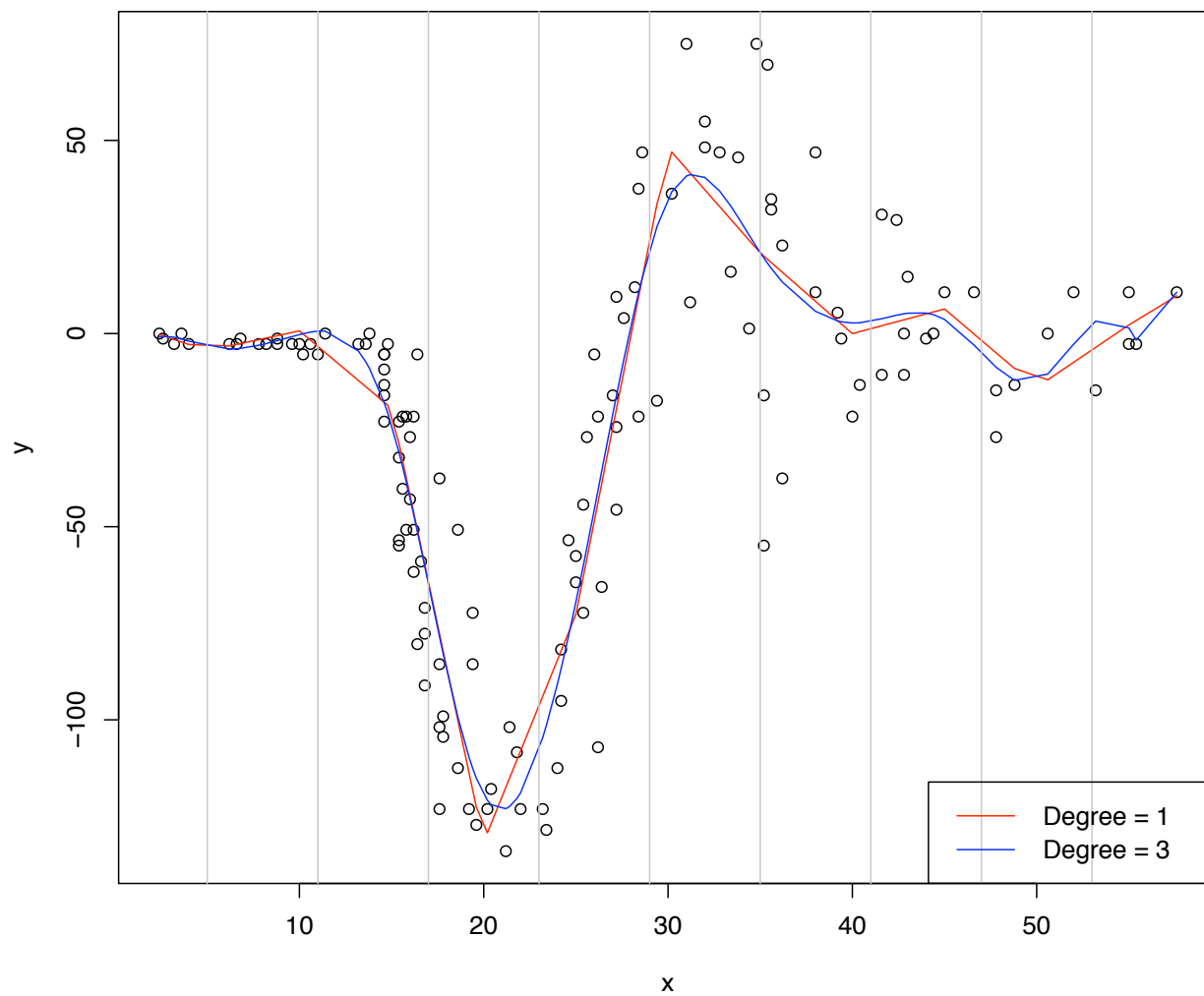
More knots  $\Rightarrow$  less smoothing

# Motorcycle data with 6 knots:





# Motorcycle data with 9 knots:



*Smoothing spline models* are defined in a slightly different way. Within a class of functions, a smoothing spline minimizes the *penalized least squares criterion*

$$\frac{1}{n} \sum_{i=1}^n (Y_i - f(X_i))^2 + \lambda \int f''(x) dx$$

The parameter  $\lambda$  controls how smooth the function is (in terms of integrated second derivative).

We can specify  $\lambda$  in terms of the equivalent *degrees of freedom* of the model, or we can choose it in a data-based way, using something called *cross validation*.

