Nonlinear Models
So far the models we have studied this semester have been linear in the sense that our model for the mean has been a linear function of the parameters.
We have assumed

$$E(Y) = X \beta$$

which is equivalent to

$$E(Y_i) = x'_i \beta \quad \text{for} \quad i = 1, \ldots, n$$

where

$$x'_i = [x_{i1}, x_{i2}, \ldots, x_{ip}]$$

is the $i$th row of $X$. 
We can write our model for the mean response as

\[ E(Y_i) = f(X_i, \beta) \]

where

\[ f(X_i, \beta) = \frac{X_i}{\beta}. \]
\[ f(x_i, \beta) = x_i'\beta \] is said to be linear in the parameters of \( \beta \) because 
\[ x_i'\beta = x_{i1}\beta_1 + x_{i2}\beta_2 + \cdots + x_{ip}\beta_p \] is a linear combination of \( \beta_1, \beta_2, \ldots, \beta_p \).

\[ f(x_i, \beta) = x_i'\beta \] is linear in \( \beta \) even if the predictor variables are nonlinear functions of other variables.
For example, if

\[ X_{i1} = 1 \]
\[ X_{i2} = \text{Amount of fertilizer applied to plot } i \]
\[ X_{i3} = (\text{Amount of fertilizer applied to plot } i)^2 \]
\[ X_{i4} = \log(\text{Concentration of fungicide on plot } i) \]

\[ f(X_i, \beta) = X_i / \beta = X_{i1} \beta_1 + X_{i2} \beta_2 + X_{i3} \beta_3 + X_{i4} \beta_4 \]
\[ = \beta_1 + \text{fert}_i \beta_2 + \text{fert}_i^2 \beta_3 + \log(\text{fung}_i) \beta_4 \]

is still linear in the parameters \( \beta_1, \beta_2, \beta_3, \beta_4 \).
Now we want to consider nonlinear models for the mean $E(y_i)$.

We want to consider cases where $f(x_i, \beta)$ cannot be written as a linear combination of $\beta_1, \beta_2, \ldots, \beta_p$. 
Why would we want to consider a non-linear model?

Pinheiro and Bates (2000) give some reasons:

- Mechanistic - based on theoretical considerations about the mechanism producing the response
- Often interpretable and parsimonious
- Can be valid beyond the range of the observed data.
Example 1:

Logistic Growth Model

\( y_i \) is height of a tree at age \( x_i \) (\( i = 1, \ldots, n \)).

\[
E(y_i) = f(x_i, \beta) = \frac{\beta_1}{1 + \exp \left( \frac{x_i - \beta_2}{\beta_3} \right)}
\]

\( y_i = f(x_i, \beta) + \varepsilon_i \) , \( E(\varepsilon_i) = 0 \), \( \text{Var}(\varepsilon_i) = \sigma^2 \)

for \( i = 1, \ldots, n \).
\[ E(Y_i) = f(x_i, \beta) = \frac{\beta_1}{1 + \exp \frac{x_i - \beta_2}{\beta_3}} \]

Note that as age \( \rightarrow \infty \), \( E(Y_i) \rightarrow \beta_1 \).

At age = \( \beta_2 \), \( E(Y_i) = \beta_1/2 \), half the asymptotic height.

\( \beta_3 \) is the time it takes a tree to grow from height \( \beta_1/2 \) to about \( 3\beta_1/4 \).

\[ (3/4 \approx \frac{1}{1+e^{-1}}) \]
Example Logistic Curves

$\beta_1=20, \beta_3=7, \beta_3=3$

$\beta_1=18, \beta_3=10, \beta_3=5$
Least Squares Estimation

\[ y_i = f(x_i, \beta) + \varepsilon_i \quad i = 1, \ldots, n \]

Find \( \hat{\beta} \) that minimizes

\[ g(\beta) = \sum_{i=1}^{n} [y_i - f(x_i, \beta)]^2. \]

We start with the same strategy used in the linear case to obtain the normal equations.
\[ \frac{d g(b)}{d b} = 0 \quad (\text{Estimating Equations}) \]

\[ \frac{d g(b)}{d b_i} = 2 \sum_{i=1}^{n} \left[ y_i - f(x_i, b) \right] \frac{d f(x_i, b)}{d b_i} \]

\[ \frac{d g(b)}{d b_p} = 2 \sum_{i=1}^{n} \left[ y_i - f(x_i, b) \right] \frac{d f(x_i, b)}{d b_p} \]
If we define

\[ f(X, \beta) = \begin{bmatrix} f(x_1, \beta) \\ \vdots \\ f(x_n, \beta) \end{bmatrix} \]

and

\[ D' = \begin{bmatrix} \frac{2f(x_1, \beta)}{db_1} & \ldots & \frac{2f(x_n, \beta)}{db_1} \\ \frac{2f(x_1, \beta)}{db_1} & \ldots & \frac{2f(x_n, \beta)}{db_1} \\ \vdots & \ldots & \vdots \\ \frac{2f(x_1, \beta)}{db_p} & \ldots & \frac{2f(x_n, \beta)}{db_p} \end{bmatrix} \]
then \( \frac{d g(b)}{d b} = 0 \) is equivalent to

\[
D'[y - f(x, b)] = 0.
\]

In the linear case, \( D' = X' \) and

\[
D'[y - f(x, b)] = 0 \quad \text{becomes}
\]

\[
x' [y - Xb] = 0 \quad \Leftrightarrow \quad X' X b = X'y.
\]
In the nonlinear case, \( D' \) depends on \( \beta \) so that the equation

\[
D'[x - f(x, b)] = 0
\]

is not so easy to solve for \( b \).
For example, for the logistic model,

\[
\frac{df(x_i, \beta)}{d\beta_1} = \frac{2}{d\beta_1} \frac{\beta_1}{1 + \exp \left\{ - (x_i - \beta_2) / \beta_3 \right\}^3}
\]

\[
= \frac{1}{1 + \exp \left\{ - (x_i - \beta_2) / \beta_3 \right\}^3}
\]

\[
\frac{df(x_i, \beta)}{d\beta_2} = -\beta_1 \frac{\exp \left\{ - (x_i - \beta_2) / \beta_3 \right\}^3}{\left[ 1 + \exp \left\{ - (x_i - \beta_2) / \beta_3 \right\} \right]^2 \beta_3}
\]
Thus, the \( i \)th column of \( D' \) is

\[
\frac{df(x_i, \beta)}{d\beta_3} = \frac{-\beta_1 \exp \left\{ -\frac{(x_i - \beta_2)}{\beta_3} \right\} (x_i - \beta_2)}{\left[ 1 + \exp \left\{ -\frac{(x_i - \beta_2)}{\beta_3} \right\} \right]^2 \beta_3^2}
\]
The Gauss Newton Algorithm for obtaining a solution to the estimating equations

\[ D'[y - f(x, k)] = 0 \]
First note that by Taylor's Theorem,

\[ f(\tilde{x}_i, b) \approx f(\tilde{x}_i, b^*) + \left[ \frac{df(\tilde{x}_i, b)}{db} \right]_{b=b^*} (b - b^*) \]

Thus,

\[ f(X, b) \approx f(X, b^*) + D^* (b - b^*) \]

where \( D^* \) is \( D \) evaluated at \( b = b^* \).
Now let $b^{(0)}$ denote an initial guess for a solution to

$$D'[x - f(x, b)] = 0$$

and let $b^{(r)}$ denote an approximation to a solution obtained at step $r$ of the iterative algorithm to be defined as follows.
Let $D^{(r)}$ denote $D$ evaluated at $b = b^{(r)}$.

Consider

$$ \left[ D^{(r-1)} \right]' \left[ \begin{array}{c} y - \frac{\sum f(X, b^{(r-1)})}{n} \\ y - D^{(r)}(b) \end{array} \right] $$

$$ = \left[ D^{(r-1)} \right]' \left[ \begin{array}{c} \frac{\sum f(X, b^{(r-1)})}{n} \\ \frac{\sum D^{(r)}(b)}{n} \end{array} \right] - D^{(r)}(b) $$
Equating to 0 we have

$$\begin{bmatrix} D^{(r-1)} \end{bmatrix}^\prime \begin{bmatrix} 3 \times -f(x, b^{(r-1)}) \end{bmatrix} - D^{(r-1)}(b - b^{(r-1)}) = 0$$

$$\iff \begin{bmatrix} D^{(r-1)} \end{bmatrix}^\prime D^{(r-1)}(b - b^{(r-1)}) = \begin{bmatrix} D^{(r-1)} \end{bmatrix}^\prime \{3 \times -f(x, b^{(r-1)})\}$$

$$\begin{bmatrix} X^* \end{bmatrix}^\prime \begin{bmatrix} X^* \end{bmatrix} \begin{bmatrix} b^* \end{bmatrix} \sim \begin{bmatrix} X^* \end{bmatrix}^\prime \begin{bmatrix} Y^* \end{bmatrix}$$

Solving for $b$, we have

$$b - b^{(r-1)} = \left(\begin{bmatrix} D^{(r-1)} \end{bmatrix}^\prime D^{(r-1)}\right)^{-1} \begin{bmatrix} D^{(r-1)} \end{bmatrix}^\prime \{3 \times -f(x, b^{(r-1)})\}$$

$$\Rightarrow b = b^{(r-1)} + \left(\begin{bmatrix} D^{(r-1)} \end{bmatrix}^\prime D^{(r-1)}\right)^{-1} \begin{bmatrix} D^{(r-1)} \end{bmatrix}^\prime \{3 \times -f(x, b^{(r-1)})\}$$
Thus, we define
\[ b^{(r)} = b^{(r-1)} + ([D^{(r-1)}]'D^{(r-1)})^{-1}[D^{(r-1)}]'\{y-f(x, b^{(r-1)})\}\]

We continue iterating until some convergence criterion is met.

For example, we could continue until
\[ g(b^{(r-1)}) - g(b^{(r)}) < \text{small constant} \]

or
\[ \max_{j=1, \ldots, p} \frac{|b_j^{(r)} - b_j^{(r-1)}|}{|b_j^{(r-1)} + \varepsilon|} < \text{small constant} \]
Selection of starting value $b^{(0)}$:

We'd like to choose a value close to the overall minimizer so that convergence will be quick and so that we won't get stuck at some local minimum.

It is often a good idea to try multiple starting values.
Normal Theory Inference:

\[ Y_i = f(x_i, \beta) + \varepsilon_i \quad i = 1, \ldots, n \]

\[ \varepsilon_1, \ldots, \varepsilon_n \sim \text{iid } N(0, \sigma^2) \]

Let \( \hat{\beta} \) be the least squares estimate of \( \beta \) (obtained by finding \( \lim_{r \to \infty} b(r) \), e.g.).

Let \[ SSE = g(\hat{\beta}) = \sum_{i=1}^{n} \left[ Y_i - f(x_i, \hat{\beta}) \right]^2 \]

Let \[ MSE = \frac{SSE}{n-p} \]
For $n$ sufficiently large,

\[
\frac{(n-p)\text{MSE}}{\sigma^2} = \frac{SSE}{\sigma^2} \sim \chi^2_{n-p}.
\]

\[\hat{\beta} \sim N(\beta, \sigma^2 (D'D)^{-1})\]

$\sigma^2 (D'D)^{-1}$ can be estimated by

\[\text{MSE} (\hat{D}'\hat{D})^{-1}, \text{ where } \hat{D} \text{ is} \]

$D$ evaluated at $\hat{\beta}$. 
An approximate F-test of $H_0: C\hat{\beta} = d$
rejects $H_0$ at level $\alpha$ if and only if

$$F = \frac{\hat{\beta}'c'[c(D'D)^{-1}c']^{-1}c\hat{\beta}/q}{\text{MSE}}$$

$$\geq F_{q, n-p}^{(\alpha)}$$

where $q = \text{rank}(c)$

$= \text{number of rows of } C$. 
An approximate $100(1-\alpha)\%$ confidence interval for $\hat{\beta}'\beta$ is

\[
\hat{\beta}' = \beta + t_{n-p} \sqrt{\text{MSE} \ \beta' (\hat{D}' \hat{D})^{-1} \beta}
\]
We also have approximate F tests for reduced vs. full model comparison:

\[
F = \frac{(SSE_{\text{reduced}} - SSE_{\text{full}})}{\text{df}_{\text{reduced}} - \text{df}_{\text{full}}} \frac{\text{SSE}_{\text{full}} / \text{df}_{\text{full}}}{\text{SSE}_{\text{full}} / \text{df}_{\text{full}}}
\]

\[H_0 \sim F_{\text{df}_{\text{reduced}} - \text{df}_{\text{full}}, \text{df}_{\text{full}}}
\]
For example, consider a test of

\[ H_0 : \beta_1 = \beta_{10} \text{ vs. } H_A : \beta_1 \neq \beta_{10} \text{ for some fixed } \beta_{10}. \]

Let \( \beta_2 = \begin{bmatrix} \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} \). Let \( f_0 (x, \beta_2) = f(x, \begin{bmatrix} \beta_{10} \\ \beta_2 \end{bmatrix}) \).

Then the reduced model is

\[ y_i = f_0 (x_i, \beta_2) + \varepsilon_i, \quad i = 1, \ldots, n \]

\( \varepsilon_1, \ldots, \varepsilon_n \overset{i.i.d.}{\sim} N(0, \sigma^2) \).
Then \( F(\beta_0) = \frac{\text{SSE}_{\text{reduced}} - \text{SSE}_{\text{full}}}{\text{MSE}_{\text{full}}} \sim F_{1, n-p} \)

The set

\[ \{ \beta_0 : F(\beta_0) \leq F(\alpha) \} \]

is an approximate 100(1-\(\alpha\))\% confidence set for \( \beta_1 \).

An equivalent set is

\[ \{ \beta_0 : -t_{n-p}^{(\alpha/2)} \leq \tau(\beta_0) \leq t_{n-p}^{(\alpha/2)} \} \]

where

\( \tau(\beta_0) = \text{sign}(\beta_0 - \hat{\beta}_1) \sqrt{F(\beta_0)} \)
Example 2:

Researchers were interested in studying the cold tolerance of different grasses.

18 plants of genotype 1 and 18 plants of genotype 2 were assigned to 6 temperature treatments using a completely randomized design with 3 plants per combination of genotype and temperature.

Following temperature treatment, a measure of ion leakage was obtained from each plant.

High ion leakage suggests that a plant sustained high damage due to the temperature treatment while low ion leakage suggests little damage.
```r
d = read.delim("http://www.public.iastate.edu/~dnett/S511/IonLeakage.txt")

d
  geno temp  y
  1    1   4 6.97
  2    1   4 5.66
  3    1   4 6.16
  4    1  -5 19.39
  5    1  -5 15.94
  6    1  -5 13.90
  7    1  -7 29.08
  8    1  -7 25.03
  9    1  -7 37.69
 10   1  -9 50.94
 11   1  -9 48.67
 12   1  -9 41.48
 13   1 -11 54.49
 14   1 -11 58.12
 15   1 -11 55.48
 16   1 -15 57.63
```
|   |   |   |      |      
|---|---|---|------|------|
| 17| 1 | -15| 61.05|
| 18| 1 | -15| 56.59|
| 19| 2 |  4 |  9.03|
| 20| 2 |  4 |  7.36|
| 21| 2 |  4 |  6.97|
| 22| 2 | -5 | 22.13|
| 23| 2 | -5 | 15.76|
| 24| 2 | -5 | 23.70|
| 25| 2 | -7 | 37.75|
| 26| 2 | -7 | 48.81|
| 27| 2 | -7 | 38.09|
| 28| 2 | -9 | 57.33|
| 29| 2 | -9 | 52.77|
| 30| 2 | -9 | 58.97|
| 31| 2 | -11| 62.85|
| 32| 2 | -11| 56.36|
| 33| 2 | -11| 61.62|
| 34| 2 | -15| 67.19|
| 35| 2 | -15| 63.55|
| 36| 2 | -15| 65.51|
d$geno=as.factor(d$geno)

plot(d$temp,d$y,col=4,pch=19,
     xlab="Temperature (degrees Celsius)",
     ylab="Measure of Ion Leakage")
Scientific considerations suggest the following 4-parameter Gompertz curve as a potentially appropriate function for the relationship between temperature and the measure of ion leakage.

\[ f(x, \beta) = \beta_1 + \beta_2 e^{-e^{-(x-\beta_3)/\beta_4}} \]
#Fit the 4-parameter Gompertz model
to the data. Initially, we will ignore the genotype factor.

\[
o = \text{nls}(y \sim b1 + b2 \cdot \exp(-\exp\left(-\left(\frac{\text{temp} - b3}{b4}\right)\right)),
\quad \text{data=d},
\quad \text{start=}\text{c}(b1=7, b2=55, b3=-5, b4=-2))
\]
Summary:

Formula: \( y \sim b_1 + b_2 \times \exp(-\exp(-((\text{temp} - b_3)/b_4))) \)

Parameters:

| Parameter | Estimate | Std. Error | t value | Pr(>|t|) |
|-----------|----------|------------|---------|----------|
| b1        | 7.1454   | 2.0089     | 3.557   | 0.00119 ** |
| b2        | 55.8565  | 3.0501     | 18.313  | < 2e-16 *** |
| b3        | -6.0301  | 0.2331     | -25.871 | < 2e-16 *** |
| b4        | -2.0950  | 0.2936     | -7.135  | 4.26e-08 *** |

Residual standard error: 4.948 on 32 degrees of freedom

Number of iterations to convergence: 5
Achieved convergence tolerance: 1.995e-06
#By default, a numerical algorithm is used to compute the derivatives needed for the Gauss-Newton algorithm. In some cases, there is an advantage to providing the derivatives along with the function. This can be done using the deriv function.

```r
f = deriv(y ~ b1 + b2 * exp(-exp(-(x - b3) / b4))), c("b1", "b2", "b3", "b4"), function(b1, b2, b3, b4, x) { }
```
f
function (b1, b2, b3, b4, x)
{
  .expr1 <- x - b3
  .expr4 <- exp(-(.expr1/b4))
  .expr6 <- exp(-.expr4)
  .value <- b1 + b2 * .expr6
  .grad <- array(0, c(length(.value), 4L),
                 list(NULL, c("b1","b2","b3","b4")))
  .grad[,"b1"] <- 1
  .grad[,"b2"] <- .expr6
  .grad[,"b3"] <- -(b2* (.expr6* (.expr4* (1/b4))))
  .grad[,"b4"] <- -b2* (.expr6* (.expr4* (.expr1/b4^2))))
attr(.value, "gradient") <- .grad
  .value
}

o=nls(y~f(b1,b2,b3,b4,temp),
   data=d,
   start=c(b1=6,b2=55,b3=-5,b4=-2))
summary(o)

Formula: y ~ f(b1, b2, b3, b4, temp)

Parameters:

| Parameter | Estimate | Std. Error | t value | Pr(>|t|)     |
|-----------|----------|------------|---------|--------------|
| b1        | 7.1454   | 2.0089     | 3.557   | 0.00119 **   |
| b2        | 55.8565  | 3.0501     | 18.313  | < 2e-16 ***  |
| b3        | -6.0301  | 0.2331     | -25.871 | < 2e-16 ***  |
| b4        | -2.0950  | 0.2936     | -7.135  | 4.26e-08 *** |

Residual standard error: 4.948 on 32 degrees of freedom

Number of iterations to convergence: 5
Achieved convergence tolerance: 1.993e-06
#Store the least squares estimator of beta.

b=coef(o)

b

\[ b \]

\[ \begin{array}{cccc}
  b1 & b2 & b3 & b4 \\
  7.145431 & 55.856466 & -6.030129 & -2.094961 \\
\end{array} \]

#Add the estimated mean curve to the plot.

x=seq(-16,6,by=.1)
lines(x,f(b[1],b[2],b[3],b[4],x))
#Examine a plot of standardized residuals versus fitted values.

plot(o)
#Get an estimate of the variance of for the squares estimator bhat:
#MSE(Dhat'Dhat)^{-1}

v=vcov(o)
round(v,3)

<table>
<thead>
<tr>
<th></th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
<th>b4</th>
</tr>
</thead>
<tbody>
<tr>
<td>b1</td>
<td>4.036</td>
<td>-4.372</td>
<td>-0.287</td>
<td>0.141</td>
</tr>
<tr>
<td>b2</td>
<td>-4.372</td>
<td>9.303</td>
<td>0.112</td>
<td>-0.590</td>
</tr>
<tr>
<td>b3</td>
<td>-0.287</td>
<td>0.112</td>
<td>0.054</td>
<td>-0.002</td>
</tr>
<tr>
<td>b4</td>
<td>0.141</td>
<td>-0.590</td>
<td>-0.002</td>
<td>0.086</td>
</tr>
</tbody>
</table>

#Obtain the error degrees of freedom.

summary(o)$df
[1] 4 32
df=summary(o)$df[2]
df
[1] 32
# Form a 95% confidence interval for beta_3.

\[ b[3] - qt(.975, df) \times \sqrt{v[3,3]} \]

\[ b_3 \]

-6.504898

\[ b[3] + qt(.975, df) \times \sqrt{v[3,3]} \]

\[ b_3 \]

-5.555361
The following function can be used to obtain approximate $100(1-a)\%$ confidence intervals for each element of $C\beta$.

ci=function(nlsout,C,a=0.05)
{
  b=coef(nlsout)
  V=vcov(nlsout)
  df=summary(nlsout)$df[2]
  Cb=C%*%b
  se=sqrt(diag(C%*%V%*%t(C)))
  tval=qt(1-a/2,df)
  low=Cb-tval*se
  up=Cb+tval*se
  m=cbind(C,Cb,se,low,up)
  dimnames(m)[[2]]=c(paste("c",1:ncol(C),sep=""),
    "estimate","se",
    paste(100*(1-a),"\% Conf.",sep=""),
    "limits")
  m
}
```r
ci(o,matrix(c(1,-1,0,0,
0,0,1,0),byrow=T,nrow=2))
```

<table>
<thead>
<tr>
<th></th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>c4</th>
<th>estimate</th>
<th>se</th>
<th>95% Conf. Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-48.71103</td>
<td>4.6992196</td>
<td>-58.283031 -39.139037</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-6.03013</td>
<td>0.2330801</td>
<td>-6.504898 -5.555361</td>
</tr>
</tbody>
</table>
Alternatively, we can use the `confint` function in R to invert reduced versus full model F tests to obtain approximate $100(1-\alpha)$% confidence intervals for each of the components of $\beta$.

```
confint(o)
Waiting for profiling to be done...
  2.5%     97.5%
 b1  3.025210  11.257025
 b2  49.865349  62.365917
 b3 -6.518746  -5.554403
 b4 -2.802860  -1.583711
```
#Now suppose we want to estimate the temperature at which the measure of ion leakage is 50.
If we equate \( f(x, \beta) \) to \( 50 \) and solve for \( x \), we get

\[
x = -\beta_4 \log \left( -\log \left( \frac{50 - \beta_1}{\beta_2} \right) \right) + \beta_3.
\]

Thus, an estimate of the temperature of interest is

\[
\hat{x} = -\hat{\beta}_4 \log \left( -\log \left( \frac{50 - \hat{\beta}_1}{\hat{\beta}_2} \right) \right) + \hat{\beta}_3.
\]
How do we obtain a standard error for $\hat{x}$?

$\hat{x}$ is a nonlinear function of $\hat{\beta}$.

One approach is to use the Delta Method.
We need to find

\[
\frac{dX}{dB} = \begin{bmatrix}
\frac{dX}{dB_1} \\
\frac{dX}{dB_2} \\
\frac{dX}{dB_3} \\
\frac{dX}{dB_4}
\end{bmatrix} = \begin{bmatrix}
\frac{\beta_4}{(50-\beta_1) \log \left( \frac{50-\beta_1}{\beta_2} \right)} \\
\frac{\beta_4}{\beta_2 \log \left( \frac{50-\beta_1}{\beta_2} \right)} \\
1 \\
-\log \left( -\log \left( \frac{50-\beta_1}{\beta_2} \right) \right)
\end{bmatrix}
\]

Let \( \hat{d} = \left. \frac{dX}{dB} \right|_{\beta = \beta} \).
The variance of \( \hat{x} \) is then approximated by

\[
\hat{\sigma}^2 \text{Var} (\hat{\beta}) \hat{\sigma} = \text{MSE} \hat{\sigma}^2 (\hat{D}' \hat{D})^{-1} \hat{\sigma}.
\]

An approximate standard error for \( \hat{x} \) is then

\[
\sqrt{\text{MSE} \hat{\sigma}^2 (\hat{D}' \hat{D})^{-1} \hat{\sigma}} = \text{SE}(\hat{x})
\]

Approximate t-based inferences would use

\[
\frac{\hat{x} - \bar{x}}{\text{SE}(\hat{x})} \sim t_{n-p}.
\]
x50 = deriv(y ~ -b4 * log(-log((50 - b1) / b2)) + b3, 
             c("b1", "b2", "b3", "b4"), 
             function (b1, b2, b3, b4) { })

x50

function (b1, b2, b3, b4)
{
  .expr2 <- 50 - b1
  .expr3 <- .expr2 / b2
  .expr4 <- log(.expr3)
  .expr6 <- log(-.expr4)
  .value <- -b4 * .expr6 + b3
  .grad <- array(0, c(length(.value), 4L), list(NULL, 
          c("b1", "b2", "b3", "b4")
  ))
  .grad[, "b1"] <- b4 * (1/b2/.expr3/.expr4)
  .grad[, "b2"] <- b4 * (.expr2/b2^2/.expr3/.expr4)
  .grad[, "b3"] <- 1
  .grad[, "b4"] <- -.expr6
  attr(.value, "gradient") <- .grad
  .value
}
x50(b[1],b[2],b[3],b[4])
b4
-8.812504
attr("gradient")

    b1    b2    b3    b4
[1,]  0.1844918  0.1415470  1  1.328127

est=as.numeric(x50(b[1],b[2],b[3],b[4]))
est
[1]  -8.812504
# Use the delta method to obtain an approximate standard error associated with the estimate.

\[
\text{der} = \text{attr}(x50(b[1], b[2], b[3], b[4]), \text{"gradient"})
\]

\[
\begin{array}{ccccc}
  & b1 & b2 & b3 & b4 \\
[1,] & 0.1844918 & 0.1415470 & 1 & 1.328127 \\
\end{array}
\]

\[
\text{se} = \text{drop}(\sqrt{\text{deriv}(\text{der}^\text{T})})
\]

\[
\begin{array}{c}
  \text{se} \\
  [1] & 0.2643512 \\
\end{array}
\]

# Obtain an approximate 95% confidence interval for the temperature at which the measure of ion leakage is 50.

\[
\hat{\text{est}} - q_t(.975, \text{df}) \times \text{se}
\]

\[
[1] & -9.35097 \\
\hat{\text{est}} + q_t(.975, \text{df}) \times \text{se}
\]

\[
[1] & -8.274038 \\
\]
# Now let's consider the genotype variable.

```r
plot(d$temp, d$y, col = as.numeric(d$geno),
    pch = as.numeric(d$geno),
    xlab = "Temperature (degrees Celsius)",
    ylab = "Measure of Ion Leakage")

legend("topright",
    c("Genotype 1", "Genotype 2"), col = 1:2, pch = 1:2)
```
# Let's fit a model that allows a separate 4-parameter Gompertz model for each genotype.

```r
ogeno = nls(y ~ b1[geno] + b2[geno] * exp(-exp(-(temp - b3[geno])/b4[geno]))),
           data = d,
           start = list(b1 = c(7, 7), b2 = c(55, 55),
                         b3 = c(-5, -5), b4 = c(-2, -2))

summary(ogeno)
```

Formula: \( y \sim b1[\text{geno}] + b2[\text{geno}] \times \exp(-\exp(-((\text{temp} - b3[\text{geno}])/b4[\text{geno}]))) \)
Parameters:

| Estimate | Std. Error | t value | Pr(>|t|) |
|----------|------------|---------|----------|
| b11 6.6605 | 2.0741 | 3.211 | 0.003309 ** |
| b12 7.7669 | 2.0879 | 3.720 | 0.000886 *** |
| b21 53.9658 | 3.3329 | 16.192 | 9.50e-16 *** |
| b22 57.4662 | 2.9887 | 19.228 | < 2e-16 *** |
| b31 -6.3494 | 0.2655 | -23.919 | < 2e-16 *** |
| b32 -5.7771 | 0.2153 | -26.838 | < 2e-16 *** |
| b41 -2.2082 | 0.3345 | -6.601 | 3.69e-07 *** |
| b42 -1.9115 | 0.2683 | -7.123 | 9.45e-08 *** |

Residual standard error: 3.628 on 28 degrees of freedom

Number of iterations to convergence: 7
Achieved convergence tolerance: 2.47e-06

b=coef(ogeno)

<table>
<thead>
<tr>
<th>b</th>
<th>b11</th>
<th>b12</th>
<th>b21</th>
<th>b22</th>
<th>b31</th>
<th>b32</th>
<th>b41</th>
<th>b42</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6.660501</td>
<td>7.766861</td>
<td>53.965787</td>
<td>57.466211</td>
<td>-6.349400</td>
<td>-5.777096</td>
<td>-2.208205</td>
<td>-1.911513</td>
</tr>
</tbody>
</table>
#Add the estimated mean curves to the plot.

```r
lines(x,f(b[1],b[3],b[5],b[7],x),col=1)
lines(x,f(b[2],b[4],b[6],b[8],x),col=2)
```
# Compare reduced versus full model.

anova(o, ogeno)
Analysis of Variance Table

Model 1: \( y \sim f(b1, b2, b3, b4, \text{temp}) \)
Model 2: \( y \sim b1[\text{geno}] + b2[\text{geno}] \times \exp(-\exp(-((\text{temp} - b3[\text{geno}])/b4[\text{geno}]))) \)

<table>
<thead>
<tr>
<th>Res.Df</th>
<th>Res.Sum Sq</th>
<th>Df</th>
<th>Sum Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td></td>
<td>783.34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>28</td>
<td>4</td>
<td>414.71</td>
<td>7.8751</td>
<td>0.0002197 ***</td>
</tr>
</tbody>
</table>
Test for a lack of fit by comparing to a cell means model with different means for each combination of genotype and temperature.

\[
\text{o} \cdot \text{cell} \text{means} = \text{lm}(y \sim \text{geno} \times \text{as.factor}(\text{temp}), \text{data} = d)
\]

\[
\text{anova(ofull, ocellmeans)}
\]

Analysis of Variance Table

Model 1: \(y \sim b1[\text{geno}] + b2[\text{geno}] \times \exp(-\exp(-((\text{temp} - b3[\text{geno}])/b4[\text{geno}]))))\)

Model 2: \(y \sim \text{geno} \times \text{as.factor}(\text{temp})\)

<table>
<thead>
<tr>
<th>Res.Df</th>
<th>Res.Sum Sq</th>
<th>Df</th>
<th>Sum Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>28</td>
<td>24</td>
<td>368.63</td>
<td></td>
<td>0.6119</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>4</td>
<td>334.51</td>
<td>34.116</td>
<td>0.6119</td>
</tr>
</tbody>
</table>