Bootstrapping

Before considering the use of bootstrapping in the context of nonlinear model parameter estimation, bootstrapping is presented more broadly as a statistical tool for performing analysis when analytical tools are not available and approximations (such as those provided by the Delta Method) may not exist or may be poor.

- There are two types of bootstrapping: nonparametric and parametric. We will focus on the former as it is the more common in practice.

Nonparametric bootstrapping: Suppose \( y_1, \ldots, y_n \) is a random sample from some population, and we wish to estimate some population parameter \( \theta \). If \( \hat{\theta} \) is an estimate of \( \theta \) with some standard error \( SE(\hat{\theta}) \), then under certain conditions a confidence interval for \( \theta \) can be formed as:

\[
\hat{\theta} \pm t^* \cdot SE(\hat{\theta}) \quad (z^* \text{ may sometimes be used instead of } t^*)
\]

The validity of this CI depends on three basic assumptions:

1. \( SE(\hat{\theta}) \) is a good estimate of the standard deviation of the sampling distribution of \( \hat{\theta} \).
2. \( \hat{\theta} \) is unbiased or nearly unbiased for \( \theta \).
3. The distribution of \( \hat{\theta} \) is approximately normal.

The method of bootstrapping can address all of these issues:

1. It can provide an estimate of \( SE(\hat{\theta}) \) when no theoretical closed-form expression for \( SE(\hat{\theta}) \) exists, or provide an alternative if we are uncertain about the accuracy of an existing estimate. For example, an approximate standard error based on the Delta Method may not be very accurate. Bootstrapping allows us to check this.
2. It can provide an estimate of the bias of \( \hat{\theta} \) as an estimator of \( \theta \).
3. It can provide information on the shape of the sampling distribution of \( \hat{\theta} \). This can be used to calculate improved confidence intervals over the normal-based ones if the sampling distribution is not normal.

So what is bootstrapping? The idea behind bootstrapping can best be explained with a simple example.

- Whenever I teach an introductory statistics course, I ask students to guess my age. Suppose for a class of 10 students, the following age guesses were given.

\[
35 \quad 41 \quad 36 \quad 34 \quad 15 \quad 38 \quad 38 \quad 43 \quad 39 \quad 82
\]

- Although we would typically summarize these age guesses using the sample mean \( \bar{y} \), we recognize the presence of some clowns in the class giving outlying age guesses. Since the mean is sensitive to the effects of outliers, we might instead use a trimmed mean \( (\bar{y}_T) \) to estimate the average age guess \( \mu \). A trimmed mean “trims off” a portion of the highest and lowest values in a set of data before computing the mean. For example, a 10% trimmed mean computes the mean from the middle 80% of the data values.
• The trimmed mean provides a compromise between the usual mean and the median, but unlike the sample mean (whose variance depends only on the population variance $\sigma^2$, $\text{Var}(\overline{y}_T)$ depends on the exact population distribution, which of course we don’t know, and hence there is no closed-form expression for this variance.

• The idea behind the bootstrap is this: if we knew the $y$-values for the entire population, then we could estimate the sampling distribution of $\overline{y}_T$ by simulation. How?

  1. 

  2. 

  3. 

We could estimate the $\text{SE}(\overline{y}_T)$ from the standard deviation of the several thousand trimmed means.

• Unfortunately, we do not know the $y$-values for the population; we only know the $y$-values for our sample. Nonparametric bootstrapping says to assume that the population looks exactly like our sample, only many times larger. This is really our best nonparametric guess as to what the population looks like.

• For example, if our $y$-values ($n = 10$) are: 

  35 41 36 34 15 38 38 43 39 82 

then we assume that 10% of the population $y$-values are 35, 10% are 41, etc. In this way, this “bootstrap population” represents our best guess as to what the actual population looks like.

• To perform bootstrapping then, we simulate drawing samples of size $n = 10$ from this “bootstrap population.” If the population is large relative to $n$, this is equivalent to drawing random samples of size 10 with replacement from the original sample of size 10. Since we are sampling with replacement, we will not necessarily get the same sample every time. For example, issuing the MatLab commands:

  ```matlab
  population = [35,41,36,34,15,38,38,43,39,82]; % Original sample of 10 values
  x = randsample(population,10,true); % Sample of 10 w/replacement
  
  two times gave the following two samples:

  15 35 15 39 35 38 15 34 39 82         39 41 82 39 38 38 38 38 38 15 43

• Finally, generate a large number of bootstrap samples (say 1000 or more), calculate the sample trimmed mean for each sample, and then calculate the sample standard deviation of the 1000 sample trimmed means as an estimate of $\text{SE}(\overline{y}_T)$.

• Efron and Tibshirani (1991) found that generally no more than 200 bootstrap samples are required to obtain a good estimate of the variance of an estimator.
Example: Finding the SE($\bar{y}_T$) using MatLab: Suppose as discussed above that a random sample of size 10 age guesses yielded the y-values 35,41,36,15,38,38,43,39, & 82, and we would like to estimate SE($\bar{y}_T$). To do this in MatLab, we could issue the commands below, and consider the following output:

```matlab
population = [35,41,36,15,38,38,43,39,82]; % Original sample of 10 values
trimpct = 20; % Total trim percentage
bootout = bootstrp(5000,@(x) trimmean2(x, ... % Takes 5000 bootstrap samples
    trimpct),population); % and computes trimmed mean
boot.true = trimmean2(population,trimpct); % True trimmed mean
boot.mean = mean(bootout); % Mean of bootstrap trimmed means
boot.bias = boot.mean-boot.true; % Bootstrap estimate of bias
boot.se = std(bootout); % Bootstrap estimate of SE

boot = true: 38 mean: 38.9224 bias: 0.9224 se: 4.3975
```

- From this output, we have the sample trimmed mean $\bar{y}_T = 38$, the bootstrap trimmed mean $\bar{y}_{TB} = 38.92$, the estimated bias ($38.92 - 38 = 0.92$), and the bootstrap estimated standard error: $SE_B(\bar{y}_T) = 4.3975$.

- The standard error of $\bar{y}$ here was $SE(\bar{y}) = s/\sqrt{n} = 5.2545$, justifying the use of a trimmed mean as a more precise measure of the center of age guesses.

- MatLab has a built-in trimmed mean function called `trimmean` which differs from my function `trimmean2` in how it handles ties in the data. Specifically, it treats similar observations as a group, rather than as individual values, so waits to trim an entire group of equivalent values instead of trimming some of them. Mine is provided below.

```matlab
function out = trimmean2(x,p)
    n = length(x);
    perc = 100*((1:n)-.5)/n;
    x = sort(x);
    newx = x((perc>=(p/2))&...               
    (perc<100-p/2));
    out = mean(newx);
```

- Issuing the commands below produces a histogram of the 5000 bootstrap values $\bar{y}_{T1}, \ldots, \bar{y}_{T5000}$ for the trimmed mean $\bar{y}_T$. What does this tell us?
Can we construct a CI of the form: $\bar{y}_T \pm t^* \cdot SE_B(\bar{y}_T)$? How can we construct a CI for the true trimmed mean?

The code below was used to construct the histogram on the previous page.

```matlab
bars = 20:0.5:60; % Defines histogram bars
hist(bootout,bars); % Histogram of bootstrap means
xlabel('Bootstrap Trimmed Means','fontsize',14, ... % Puts x-label on plot in
'fontweight','b') % bold w/ font size 14
ylabel('Frequency','fontsize',14,'fontweight','b') % Puts y-label on plot
title('Histogram of Bootstrap Trimmed Means', ... % Puts title on plot
'fontsize',14,'fontweight','b')
```

- The sampling distribution appears to be multimodal (NOT NORMAL!) due mainly to the large outlying age guess. There are methods for obtaining confidence intervals which adjust for the bias and the nonnormal sampling distribution. One of the methods built into MatLab is the so-called Bias-Corrected & Accelerated (BCa) method:

```matlab
boot.ci = bootci(5000,{@(x) trimmean2(x, ... % Computes BCa bootstrap 95%
trimpct),population},'alpha',.05); % CI for trimmed mean
boot.ci = 31.6250 49.7500
```

- A 95% nonparametric CI based on this method is (31.625, 49.750), which is not symmetric about the estimate of 38. The BCa method is based on the actual distribution of the bootstrap estimates (rather than just their standard deviation) and requires a large number of bootstrap replications to be reliable (say 5000 or more for a 95% CI). In general, more bootstrap replications are needed to reliably estimate percentiles further in the tails (for a 99% CI, for example, where one needs the .5% and 99.5% percentiles).

Some Background on Bootstrapping

- When did bootstrapping originate? The first paper on the theory of bootstrapping is attributed to Bradley Efron (1979), but it wasn’t until the early 1990’s with the Science paper by Efron & Tibshirani (1991), their subsequent book (1993), and advances in modern computing that the method gained widespread attention and use. It is now the most common method of obtaining measures of uncertainty in numerous statistical applications where no closed-form expressions and/or approximations are available.

- Why is it called bootstrapping? The term “bootstrapping” stems from a quote in the 1786 book *Singular Travels, Campaigns, and Adventures of Baron Munchausen* by R.E. Raspe, a collection of very TALL tales and adventures of the fictional character Baron Munchausen. The quote of interest is:
I was still a couple of miles above the clouds when it broke, and with such violence I fell to the ground that I found myself stunned, and in a hole nine fathoms under the grass, when I recovered, hardly knowing how to get out again. Looking down, I observed that I had on a pair of boots with exceptionally sturdy straps. Grasping them firmly, I pulled with all my might. Soon I had hoist myself to the top and stepped out on terra firma without further ado.

- So the idea of resampling your data to generate more data is akin to “pulling yourself up by your bootstraps” to do analysis. Statisticians were very suspicious of bootstrapping when it first gained attention, but are now convinced of its power. If it seems like you are getting something for nothing (or downright cheating) with bootstrapping, I tend to agree, but the method has proven useful in an incredible variety of problems (censored data, survival data, time series, classification trees, goodness of fit statistics, linear and nonlinear regression, etc.).

**Bootstrapping in Statistical Models:** Bootstrapping is used to estimate standard errors of parameter estimates in statistical models in three basic ways, depending on how you view the data and the assumptions you are willing to make. For illustration, suppose there is a response variable $y$ being explained by $k$ explanatory variables $x = (x_1, \ldots, x_k)$ using $p$ parameters $\theta = (\theta_1, \ldots, \theta_p)$:

$$y_i = f(x_i|\theta) + \epsilon_i, \quad i = 1, \ldots, n,$$

or

$$\begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_n
\end{bmatrix} = 
\begin{bmatrix}
f(x_{1i}, \theta) \\
f(x_{2i}, \theta) \\
\vdots \\
f(x_{ni}, \theta)
\end{bmatrix} + 
\begin{bmatrix}
\epsilon_1 \\
\epsilon_2 \\
\vdots \\
\epsilon_n
\end{bmatrix}.$$

where $x_i' = [x_{1i}, x_{2i}, \ldots, x_{ki}]$ is the row vector of observations on $k$ explanatory variables $x_1, \ldots, x_k$ for the $i^{th}$ observational unit, and $n$ is the number of observations.

- Recall that if $f(\theta) = X\theta$, this is the general linear model (GLM) discussed previously. Otherwise, it is nonlinear in the model parameters $\theta$.

1. **Bootstrap $(x, y)$ pairs:** If we consider the $y$’s and $x$’s to be jointly random quantities from some multivariate distribution $g(y, x)$, then bootstrapping will take place by resampling pairs of the original $(y, x)$ values.

2. **Bootstrap $y$ given $x$:** If we consider the $x$’s to be fixed quantities (the common regression assumption) so that the $y$’s have conditional distribution given by $g(y|x)$, then bootstrapping will take place by resampling residuals from the fitted model at the observed (fixed) $x$-values.

3. **Parametric bootstrap:** Considering the $x$’s as fixed values, generate $y$-values for a given $x$ using the fitted distribution. In this case, we are not resampling the data; we are using the fitted model to generate new data.
Example: Recall the bighorn sheep data (shown to the right) from the nonlinear model estimation notes. In those notes, we discussed the use of the Delta Method to approximate the standard errors of the parameter estimates in the nonlinear logistic growth model of the form:

\[ y_i = \frac{\hat{M}u}{\hat{u} + (\hat{M} - \hat{u}) \exp\{-\hat{\gamma}kt_i\}} + \epsilon_i, \]

where \( \hat{M} = 207.6399, \hat{u} = 67.9731, \hat{k} = 0.1380, \hat{\gamma} = 3.4542 \), as computed using the Gauss-Newton method outlined earlier.

- The Delta Method standard errors for these three parameters were found to be:

\[
\begin{align*}
\text{SE}_{DM}(\hat{M}) &= 4.178, \\
\text{SE}_{DM}(\hat{u}) &= 7.166, \\
\text{SE}_{DM}(\hat{k}) &= 0.0091, \\
\text{SE}_{DM}(\hat{\gamma}) &= 0.8092.
\end{align*}
\]

- Here, the goal is to show how bootstrapping can be used to estimate the standard errors and check the accuracy of the Delta method standard errors above.

**Method 1 (Nonparametric bootstrap 1)**

1. Take a sample of \((x, y)\)-pairs of size \(n = 17\) with replacement from the 17 pairs of observed data values \((x_1, y_1), (x_2, y_2), \ldots, (x_{17}, y_{17})\).
2. Refit the logistic growth model using least squares to obtain parameter estimates for the resampled data: \(\hat{\theta} = (\hat{M}, \hat{u}, \hat{k}, \hat{\gamma})\).
3. Repeat steps 1 & 2 many times (say \(B = 5000\) times) to generate 5000 bootstrap samples and corresponding sets of parameter estimates \(\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_B\).
4. The bootstrap standard errors for the parameter estimates are estimated as the standard deviation of the \(B = 5000\) parameter estimates.

To illustrate Method 1 in MatLab, you can either use the `bootstrp` function or write a loop. Both are shown below. First, we fit the generalized logistic model to these data.

```matlab
nlin_fn = @genlogistic; % Defines the nonlinear model
[betahat,resid,J] = nlinfit(time, ... % Performs NLS, returning the parameter
sheep82.count,nlin_fn,beta); % estimates, residuals, & Jacobian
mse = sum(resid.^2)/(n-p); % Computes the mean squared error (MSE)
```
se.delta = sqrt(diag(inv(J'*J).*mse)); % Computes the Delta Method SEs
B = 5000; % Number of bootstrap samples

% This line computes bootstrap estimates of the model parameters using the % bootstrap function directly, where (x,y) are resampled as pairs.

betaboot = bootstrp(B,@(x,y) nlinfit(x,y,nlin_fn,betahat), ... % OR
    time,sheep82.count);

for i = 1:B % Loops through bootstrap samples
    bsamp = randsample(1:n,n,true); % Samples n pairs w/replacement
    xboot = time(bsamp); % Times for the n pairs
    yboot = sheep82.count(bsamp); % No. counts for the n pairs
    betaboot(i,1:p) = nlinfit(xboot, ... % ith bootstrap parameter estimates
        yboot,nlin_fn,betahat); % stored as 4 columns in row i
end % End of loop

boot.true = betahat; % True parameter estimates
boot.mean = mean(betaboot); % Mean of bootstrap estimates
boot.bias = boot.mean-boot.true; % Bootstrap estimate of bias
boot.se = std(betaboot); % Bootstrap estimate of SE

This results in the following table of bootstrap estimates:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Bootstrap Mean</th>
<th>Bootstrap Bias</th>
<th>Bootstrap Standard Error</th>
<th>Delta Method Standard Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>207.6399</td>
<td>208.2120</td>
<td>0.5721</td>
<td>6.1064</td>
<td>4.178</td>
</tr>
<tr>
<td>u</td>
<td>67.9731</td>
<td>67.9061</td>
<td>-0.0670</td>
<td>5.6245</td>
<td>7.166</td>
</tr>
<tr>
<td>k</td>
<td>0.1380</td>
<td>0.1403</td>
<td>0.0023</td>
<td>0.0404</td>
<td>0.0091</td>
</tr>
<tr>
<td>γ</td>
<td>3.4542</td>
<td>3.4613</td>
<td>0.0071</td>
<td>0.7359</td>
<td>0.8092</td>
</tr>
</tbody>
</table>

Comparing the two estimates of standard errors in the above table, what can we say about the bootstrap standard errors? The distributions of the bootstrap estimates of the parameters are shown below along with basic bootstrap summary information, as given by the function bootoutput.m. This function can be found on the course webpage with these notes.

parnames = {'M';'u';'k';'gam'}; % Parameter names
out.meth1 = bootoutput(betaboot,betahat,parnames,.05); % Function to produce
        % standard bootstrap output

>> out.meth1

72
ans = true: [207.6399 67.9731 0.1380 3.4542]
mean: [208.2120 67.9061 0.1403 3.4613]
bias: [0.5721 -0.0670 0.0023 0.0071]
se: [6.1064 5.6245 0.0404 0.7359]
cilow: [197.5834 60.8882 0.1270 2.2655]
ciupp: [221.2604 80.6218 0.1577 5.1223]

Note that the distribution of $\hat{k}$ has a tremendously long upper tail, with some values as large as 3.5. The higher bootstrap SE for this quantity results from the occasional case where none of the earlier x-values are included in the bootstrap sample. The bootstrap distributions for $\hat{M}$, $\hat{u}$ and $\hat{\gamma}$ are all slightly right-skewed, indicating the Delta Method may be slightly off for these parameters.

- In the process of performing the 5000 bootstrap nonlinear model fits, the MatLab message below appeared indicating the instability in one case due to an insufficient number of resampled pairs in the tail of the curve.

```
Warning: The Jacobian at the solution is ill-conditioned, and some model parameters may not be estimated well (they are not identifiable). Use caution in making predictions.
```

- It seems the BCa-based confidence intervals can only be computed in MatLab for a one-dimensional statistic; thus, it cannot be used to provide CIs for 4 parameters in a model setting as in this case. As an alternative, simple percentiles can be used to construct a
95% confidence interval for the model parameters as given by the \texttt{bootoutput} function. This method is considered poor if the distribution is severely skewed, and the BCa method is generally the preferred choice.

\begin{verbatim}
M   u    k    gam  
---  ---  ---   ---
\texttt{ci.low} = 197.6  60.89  .1270  2.266
\texttt{ci.upp} = 221.3  80.62  .1577  5.122
\texttt{betahat} = 207.6  67.97  .1380  3.454
\end{verbatim}

- Note that these confidence intervals are asymmetric about the parameter estimates, especially that for \( k \).

\textbf{Method 2 (Nonparametric bootstrap 2 - Fixed \( x^\prime \)s)}: Before giving the steps to carry out this second type of bootstrap, some background theory on the distribution of the residuals is needed.

- Recall that for linear models, we derived the variance-covariance matrix of the parameter vector \( \hat{\beta} \) as: \( \text{Var}(\hat{\beta}) = (X'X)^{-1}\sigma^2 \), where \( \sigma^2 \) is the error variance in the model (that is: \( \text{Var}(\epsilon_i) = \text{Var}(y_i|x_i) = \sigma^2 \)).

- Using the facts that \( \hat{y} = X\hat{\beta} \) and \( \hat{\epsilon} = y - \hat{y} \), it can be shown that:
  \[
  \text{Var}(\hat{\epsilon}) = [I - X(X'X)^{-1}X']\sigma^2.
  \]
  Hence, letting \( h_{ii} \) represent the \( i^{th} \) value on the diagonal of \( X(X'X)^{-1}X' \), we can write the variance of the \( i^{th} \) residual as: \( \text{Var}(\hat{\epsilon}_i) = (1 - h_{ii})\sigma^2 \).

- The \( n \) diagonal values \( h_{11}, \ldots, h_{nn} \) of \( X(X'X)^{-1}X' \) are called the leverage values of the model. The larger the value of \( h_{ii} \), the more leverage or influence the \( i^{th} \) data value has on the model estimates.

- Recall that one assumption made on the errors (\( \epsilon_i \)'s) in both linear and nonlinear models is that the variance is constant for each fixed \( x \)-value. The more different the \( h_{ii} \)-values are from one another, the more evidence there is of heteroscedasticity or non-constant variance in the residuals.

- Recalling that the \( \hat{F} \)-matrix of partial derivatives of \( f(\theta) \) with respect to \( \theta \) in nonlinear models plays the role of the design matrix \( X \) in linear models, the leverages \( h_{11}, \ldots, h_{nn} \) in nonlinear models are also just the diagonals of the \( \hat{F}(\hat{F}'\hat{F})^{-1}\hat{F}' \) matrix.

- To understand what these leverage values are telling us, consider the \( n = 17 \) leverages computed via MatLab for the years 1965-1982 for the bighorn sheep data:

\begin{verbatim}
[betahat,resid,J] = nlinfit(time, ... % Performs NLS, returning the parameter
sheep82.count,nlin_fn,beta); % estimates, residuals, & Jacobian
leverage = diag(J*inv(J'*J)*J'); % Computes leverage values
leverage = 0.4235 0.2908 0.2059 0.1870 0.2603 0.3422 0.3324 0.3836 0.3957 0.1982 0.1177 0.1328 0.1440 0.1463 0.1465 0.1465
\end{verbatim}
• Note that as the years increase, the leverages tend to decrease. This indicates that
data corresponding to the initial population of the generalized logistic growth model
have more influence on the fitted model than the data from the later years. This may
also explain why the \( u \)-parameter exhibits more variability than the \( M \)-parameter in
its estimate. Note also that the residual variances range from \( \text{Var} (\hat{\epsilon}_1) = (1 - .424)\sigma^2 = 
0.576\sigma^2 \) to \( \text{Var} (\hat{\epsilon}_{12}) = (1 - .147)\sigma^2 = 0.853\sigma^2 \), so that some variances are 1.5 times the
size of others.

• Due to this heteroscedasticity in the residuals, we may need to adjust the residuals in
developing a bootstrap procedure that resamples \( y \)-values for the fixed \( x \)-values from
the fitted model. The procedure, which effectively bootstraps the residuals, follows:

1. For each fixed \( x_i \), compute the corresponding predicted \( \hat{y}_i = f(x_i, \hat{\theta}) \) from the fitted
model, \( i = 1, 2, \ldots, n \).

2. For each predicted \( \hat{y}_i \), compute the corresponding residual \( \hat{\epsilon}_i = y_i - \hat{y}_i \) for \( i = 1, 2, \ldots, n \).

3. To correct for the potential heteroscedasticity in the residual variances, compute the
modified residuals: \( r_i = \hat{\epsilon}_i / \sqrt{1 - h_i} \) and compute the centered modified residuals
\( r_i - \bar{r} \) for \( i = 1, 2, \ldots, n \).

4. Take a sample of the \( n \) modified and centered residuals with replacement.

5. Add these resampled residuals to the fitted \( \hat{y}_i \)-values to generate a bootstrap sample of
\( y \)-values corresponding to the original fixed \( x \)-values.

6. Refit the growth model using least squares to obtain parameter estimates for the re-
sampled data: \( \hat{\theta} = (\hat{M}, \hat{u}, \hat{k}, \hat{\gamma}) \).

7. Repeat steps 4, 5, & 6 many times (say \( B = 2000 \) times) to generate 2000 bootstrap
samples and corresponding sets of parameter estimates \( \hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_B \).

8. The bootstrap standard errors for the parameter estimates are estimated as the standard
deviations of the \( B = 2000 \) parameter estimates.

Method 2 is illustrated in MatLab as follows:

\[
\begin{align*}
\text{yhat} & = \text{sheep82.count-resid;} \quad \% \text{Computes predicted counts (y-hats)} \\
\text{leverage} & = \text{diag(J*inv(J'*J)*J')} \quad \% \text{Computes leverage values} \\
\text{yhatmat} & = \text{ones(B,1)*yhat'} \quad \% \text{Creates a B x n matrix of} \\
& \quad \% \text{y-hats, repeated row by row.} \\
\text{modres} & = \text{resid./sqrt(1-leverage)} \quad \% \text{Computes modified residuals.} \\
\text{modres} & = \text{modres-mean(modres)} \quad \% \text{Centers modified residuals.} \\
\text{residmat} & = \text{randsample(modres,B*n,true)} \quad \% \text{Samples the vector of} \\
& \quad \% \text{modified residuals Bxn times, with replacement.} \\
\text{residmat} & = \text{reshape(residmat,B,n)} \quad \% \text{Puts resids in B x n matrix} \\
\text{bsamp} & = \text{yhatmat + residmat;} \quad \% \text{Creates B sets of new y’s}
\end{align*}
\]
for i=1:B
    betaboot(i,1:p) = nlinfit(time, ... % Performs NLS, returning the
    bsamp(i,:)’,nlin_fn,betahat);
end % End of for loop

out.meth2 = bootoutput(betaboot,betahat,parnames,.05); % Function to
% produce standard bootstrap output

The findings from the various methods for finding std. errors are summarized below:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>(Delta Method)</th>
<th>(Bootstrap 1)</th>
<th>(Bootstrap 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>207.640</td>
<td>4.1782</td>
<td>6.4838</td>
<td>4.0546</td>
</tr>
<tr>
<td>$u_0$</td>
<td>67.973</td>
<td>7.1657</td>
<td>5.5641</td>
<td>6.8605</td>
</tr>
<tr>
<td>$k$</td>
<td>0.138</td>
<td>0.0091</td>
<td>0.0770</td>
<td>0.0095</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>3.454</td>
<td>0.8092</td>
<td>0.7232</td>
<td>0.8799</td>
</tr>
</tbody>
</table>

Note that the standard error for $\hat{k}$ is considerably smaller with this method than with the first bootstrap method. Why?
A third technique for bootstrapping entails making an assumption on the distribution of the residuals resulting from the model fit and simulating data based on the assumed residual model (typically normal) and the fitted model. This is known as **parametric bootstrapping**.

For nonlinear models, the errors are rarely normal and are generally difficult to characterize, so that the common recommendation is to not use this technique for nonlinear models. Some also feel that this violates the “spirit of bootstrapping” in that it doesn’t actually resample the data.

Davison & Hinkley (1998) recommend the use of the nonparametric bootstrap method 2 outlined above. Their recommendation is based on two main issues, one intuitive and one empirical:

1. By fixing the $x$-values, this more closely mimics the assumption of fixed $x$-values made in standard regression analysis.
2. Secondly, we are more likely to achieve the same degree of coverage in the $x$-values as in our original data. This was absolutely the problem with the high standard errors in estimating $k$ in the logistic model with Method 1 above.