There is an R code set at the end of this lab and posted on the course web page that will prove useful for doing this lab. **Use it in answering/doing the following.**

1. Consider again the "Stackloss/Nitrogen Plant" data.

a) Make the Analysis of Variance Table (exactly as described in lecture) for testing the hypothesis $H_0: \beta_2 = \beta_3 = 0$ in a model that includes all of $x_1, x_2, x_3$ as predictors. What is a corresponding $p$-value? Interpret this value.

b) Identify the two "best" (in terms of $R^2$) models of "sizes" (numbers of predictors) 1 and 2 in this example using a graphical display.

c) Set the R seed to 0 inside the `cv.lm()` call and compare the best (in terms of $R^2$) models of sizes 1, 2, and 3 in terms of cross-validation error sums of squares using first 5-fold cross-validation and then LOO (leave one out) cross-validation. What model seems "best" for prediction purposes.

d) This data set is pretty small. One might expect that cross-validation results are somewhat "noisy" because of this fact. Set the R seed inside the `cv.lm()` call to 17 and redo the 5-fold cross-validations. How much does the analysis and outcome change? (If there are big changes, one might contemplate redoing this a number of times and averaging CV error sums of square when the data set is small.)

2. Consider a logistic regression analysis of the Challenger as provided in class. (From what I was able to find online, the ambient temperature at launch was 36°F.)

a) Using the `glm()` function, what is output as "predictions"? How does one translate them to success probabilities?

b) How are the 95% confidence limits provided by R for the model parameters consistent with Figure A.21 of the text?

c) What (according to the fitted model) was the probability of an O-ring failure at the launch temperature?

3. Now consider a non-linear regression problem. The classic textbook *Statistics for Experimenters* by Box, Hunter and Hunter contains a small data set taken originally from the paper "Evaluation of methods for estimating biochemical oxygen demand parameters" by Marske and Polkowski (1977, *J. Water Pollut. Control Fed.*, **44**, No. 10, 1987). It is reproduced below. $y$ is BOD (biochemical oxygen demand) in mg/liter, a measure of pollution. A small amount of domestic or industrial waste is
mixed with pure water, sealed, and incubated for a number of days, \( x \). The data represent values obtained from \( n = 6 \) different bottles.

<table>
<thead>
<tr>
<th>( x )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y )</td>
<td>109</td>
<td>149</td>
<td>149</td>
<td>191</td>
<td>213</td>
<td>224</td>
</tr>
</tbody>
</table>

Apparently the original investigators considered the model

\[ y_i = \beta_1 (1 - \exp(-\beta_2 x_i)) + \epsilon_i \]

(for iid mean 0 normal errors \( \epsilon_i \)) to be potentially useful here. Note that in this model, \( \beta_1 \) is a limiting BOD and \( \beta_2 \) is a rate constant that controls how fast this limiting BOD is approached. For example, when \( \beta_2 x = 1 \), the model says that mean BOD is at a fraction \( (1 - \exp(-1)) = .63 \) of its limiting value.

a) Plot \( y \) vs \( x \) and make "eye-estimates" of the parameters based on your plot and the interpretations of the parameters offered above. (Your eye-estimate of \( \beta_1 \) is what looks like a plausible limiting value for \( y \) and your eye-estimate of \( \beta_2 \) is the reciprocal of a value of \( x \) at which \( y \) has achieved 63% of its maximum value.)

b) Use \texttt{nls()} and fit the model above via least squares. What are the least squares estimate of the parameter vector and the "deviance" (error sum of squares)

\[
b_{\text{OLS}} = \left( \begin{array}{c} b_1 \\ b_2 \end{array} \right) \quad \text{and} \quad \text{SSE} = \sum_{i=1}^{6} \left( y_i - \left( b_1 (1 - \exp(-b_2 x_i)) \right) \right)^2 
\]

c) Re-plot the original data with a superimposed plot of the fitted equation.

d) The time, say \( t_{100} \), at which mean BOD is 100 mg/l is a function of \( \beta_1 \) and \( \beta_2 \). Find a sensible point estimate of \( t_{100} \).

e) As a means of visualizing what function the \texttt{R} routine \texttt{nls} minimized in order to find the least squares coefficients, make a contour plot of the sum of squares function. As a check to see that you have the function programmed correctly, evaluate this function at \( b_{\text{OLS}} \) for the data in hand and verify that you get the error sum of squares (the "deviance"). (I tried using vector calculations here, but ran into problems that I don't understand trying to evaluate the sum of squares over a grid of parameter values. The above will work. You are welcome to try something slicker if you like, but don't ask me to tell you why you get error messages, if you do!) This is a "topographic map" of the sum of squares surface.
f) Identify on your plot the "Beale" 90% approximate confidence region for the parameter vector \((\beta_1, \beta_2)\)

This is the set of parameter vectors with corresponding residual sum of squares below

\[
deviance \times \left(1 + \frac{2}{6-2} F_{2,6-2}\right)
\]

where the F value is the upper 10% point.

g) Say how the R confidence limits for the \(\beta\)'s are or are not consistent with your answer to f).

---

**Code Set for Stat 401B Laboratory #8**

```r
# Here is code for Stat 401B Lab 8
# First a bit more on "Ordinary" Multiple Linear Regression
# For the Nitrogen Plant Data
stackloss
x1=stackloss[,1]
y=stackloss[,4]
x2=stackloss[,2]
x3=stackloss[,3]

# Here is a way to get the information for a partial F test directly
anova(lm(y~x1,data=stackloss),lm(y~x1+x2+x3,data=stackloss))

# Add the leaps package
# Now compute and plot a graphic identifying the 2 models
# of each size with the largest R^2

a<-regsubsets(y~.,nbest=2,data=stackloss[,1:3])
plot(a,scale="r2")
```

#Now try some Cross-Validations looking for good models

#First add the DAAG package

#Now do some cross-validations using first m=5 folds and then m=21 folds

set.seed(0)
dt = data.frame(x1,x2,x3,y)
a<-cv.lm(dt,y~x1+x2+x3,m=5,seed=0)
a

t((a$cvpred-dt$y))%*(a$cvpred-dt$y)

a<-cv.lm(dt,y~x1+x2,m=5,seed=0)
a

t((a$cvpred-dt$y))%*(a$cvpred-dt$y)

a<-cv.lm(dt,y~x1,m=5,seed=0)
a

t((a$cvpred-dt$y))%*(a$cvpred-dt$y)

set.seed(0)
a<-cv.lm(dt,y~x1+x2+x3,m=21)
a

t((a$cvpred-dt$y))%*(a$cvpred-dt$y)

a<-cv.lm(dt,y~x1+x2,m=21)
a

t((a$cvpred-dt$y))%*(a$cvpred-dt$y)
a<-cv.lm(dt,y~x1,m=21)
a
t((a$cvpred-dt$y))%*%(a$cvpred-dt$y)

#Now do some logistic regressions

#First make a data frame for the Challenger Data

Challenger<-
data.frame(y=c("N","Y",rep("N",6),rep("Y",3),rep("N",2),"Y",rep("N",6),"Y","N","Y"),temp=c(66,70,69,68,67,72,73,70,57,63,70,78,67,53,67,75,70,81,76,79,75,76,58))
Challenger

#Load the stats package, then carry on

challenger.out<-glm(y~temp,data=Challenger,family=binomial)
challenger.out
summary(challenger.out)

challenger.out$coefficients[1]
challenger.out$coefficients[2]

confint(challenger.out)

predict(challenger.out,se.fit=TRUE)
invlogit<- function (u) {exp(u)/(1+exp(u))}
invlogit(predict(challenger.out))
p <- function (x)
{1/(1+exp(-(challenger.out$coefficients[1] +
    challenger.out$coefficients[2]*x)))}

curve(p,30,90,ylim=c(0,1))
y <- c(Challenger$y) - 1
points(temp, y)
p(seq(30, 90, 10))
p(36)

# Finally, do some non-linear regression
# First make a BOD data frame
BOD <- data.frame(x=c(1, 2, 3, 5, 7, 10), y=c(109, 149, 149, 191, 213, 224))
BOD

# Now fit the nonlinear model
BOD.out <- nls(y ~ b1*(1-exp(-b2*x)), data=BOD, start=c(b1=200, b2=1), trace=T)
BOD.out
summary(BOD.out)
confint(BOD.out)

# Plot the fitted function and the data
bio <- function (x)
{coef(BOD.out)[1]*(1-exp(-coef(BOD.out)[2]*x))}
curve(bio, 0, 11, ylim=c(0, 250))
points(BOD$x,BOD$y)

# Make a contour plot of the SS surface
ss<-function (b1,b2)
{
  (109-b1*(1-exp(-b2*1)))^2+
  (149-b1*(1-exp(-b2*2)))^2+
  (149-b1*(1-exp(-b2*3)))^2+
  (191-b1*(1-exp(-b2*5)))^2+
  (213-b1*(1-exp(-b2*7)))^2+
  (224-b1*(1-exp(-b2*10)))^2
}

beta1<-150:300
beta2<-seq(.2,1.4,.01)
SumofSquares<-outer(beta1,beta2,FUN=ss)
contour(beta1,beta2,SumofSquares,levels=seq(1200,6200,1000))