```
#Analysis of the space shuttle Challenger data (see course slides for details)

challenger = read.csv("challenger.csv", header = T)

#change names to lower case
names(challenger)=c("temp", "failure")

#create indicator variable: zero if no failure and 1 if failure
challenger$fail01 = rep(0, 24)
challenger$fail01[challenger$failure=="Yes"] = 1

#let's see what a plot of temperature vs. fail01 looks like
plot(challenger$fail01, x = challenger$temp, xlab = "Temperature", ylab = "Failure")

#this is not easy to interpret.  We can try a box plot...

boxplot(challenger$temp~challenger$fail01, ylab = "Temperature", xlab = "Failure")

#this doesn't really portray a prediction of failure from temperature, but it's somewhat
useful to get an overall sense of the data.

#let's mean-center the temperature
challenger$tempcent = challenger$temp - mean(challenger$temp)

#let's fit a logistic regression to the mean centered temp
logreg = glm(fail01 ~ tempcent, family = binomial, data= challenger)

#MAKE SURE YOU USE THE glm COMMAND, NOT THE lm COMMAND.
#AND MAKE SURE TO INCLUDE THE family = binomial.

#look at results
summary(logreg)

#confidence intervals for the coefficients
confint.default(logreg)

#here is how to get the predicted probabilities for the observed cases
predprobs = predict(logreg, type = "response")

#useful to examine a plot of predicted probabilities by X
plot(y=predprobs, x = challenger$tempcent, xlab = "Temperature (Centered)", ylab = "Predicted
Probability of Failure")

#can show on original scale, too
plot(y=predprobs, x = challenger$temp, xlab = "Temperature", ylab = "Predicted Probability of
Failure")

#you might want to make a graph with more values of temperature. You have to predict new
observations -- more on that later.

#predicted probabilities at new temperatures, say 36 degrees and 68 degrees

newdata = challenger[1:2,]
newdata$temp[1] = 36
newdata$tempcent[1] = 36 - mean(challenger$temp)
newdata$temp[2] = 68
newdata$tempcent[2] = 68 - mean(challenger$temp)

predict(logreg, newdata, type="response", se.fit=T)

#you can get 95% prediction intervals for the probabilities by using 1.96 as a multiplier
#for 68 degrees, the 95% prediction interval for the probability of failure is (.315 -
1.96*.1118, .315 + 1.96*.1118)
```

```
### DIAGNOSTICS  #####

#let's look at raw residuals

rawresids = challenger$fail01 - predprobs

plot(y=rawresids, x=challenger$tempcent, xlab = "Temperature (centered)", ylab = "Residuals")

#raw residuals are not very useful!!  Can look to see which cases have values near 1 and -1
to look for
#cases that don't fit well, but not too useful otherwise

###binned residuals -- used like residual plots in linear regression.

#note: binned plots don't work so well on small sample sizes, like these data. So, the code
below
#is mostly intended for the syntax.  the script "Interpreting binned plots" shows a better
example.
#Note: in class we switched to that script at this point.

#install the arm package in R to use the binnedplot command.
install.packages("arm")
library(arm)

#pick number of classes so you have a decent sample size in each class.
#you can let the binnedplot command pick the number of classes, since it has sensible
defaults

#plot versus predicted probabilities. useful as a "one-stop shopping" plot.
# useful when many X variables and you want an initial look at model adequacy

binnedplot(x= predprobs, y= rawresids, xlab = "Predicted Probabilities")

#also can plot versus individual predictors
binnedplot(x= challenger$tempcent, y= rawresids,  xlab = "Binned Temperatures (centered)")

#there are so few data points here that it is hard to judge the quality of the plots.
#really you want at least 100 data points before these plots start being useful...

#note: you can use the binnedplot command for exploratory data analysis, too!  Just input the
outcome variable for y = ..., and the predictor variable for x = ....
#we will see this in the analysis in the script, "logistic regression 2"
#this command is only useful for continuous predictors -- if you have categorical predictors
use the tapply command (see R script for "logistic regression 2")
#when using the plot for exploratory purposes, ignore the SE lines -- they are not valid when
using the outcome

### here is how to make a Confusion Matrix
#first select the threshold for the predicted probabilities.
#Above the threshold, you would predict that they are are 1
#you can try any threshold you want -- just change the value of "threshold"

threshold = 0.5
table(challenger$fail01, logreg$fitted > threshold)

#     FALSE TRUE
#  0    16    1
#  1     4    3

#in the output, the 0 row corresponds to true y_i = 0, and the 1 row corresponds to true y_i
= 1
#the FALSE column corresponds to predicted probabilities less than the threshold
#the TRUE column corresponds to predicted probabilities above the threshold
#ideally most of the count is on the diagonal, which is what we see here.
```

```
#sensitivity -- true positive rate -- at 0.5 threshold
#3 / (3 + 4)

#specificity -- true negative rate -- at 0.5 threshold
16 / (16 + 1)

#1 - specificity is the false positive rate.  1 - 16/17 = 1/17

##ROC curve -- plots sensitivity vs 1 - specificity for an expansive set of thresholds
# first install the pROC package
install.packages("pROC")
library("pROC")

roc(challenger$fail01, fitted(logreg), plot=T, legacy.axes=T)

#can add the "best" threshold to the graph (one with highest sum of sensitivity and
specificity)

roc(challenger$fail01, fitted(logreg), plot=T, print.thres="best", legacy.axes=T)
```