Regression models in R

This weeks hands-on exercises draw from a nice set of introductory R notes written by John Verzani, entitled "simpleR – Using R for Introductory Statistics." This text used to available online at: `http://www.math.csi.cuny.edu/Statistics/R/simpleR/` (the page still exists, but the PDF is not available as of Sept. 2011, but see below).

I've put up a PDF version of the Verzani text on GitHub. Also from the class wiki download the set of R functions and data that are associated with the Verzani text `simpleR.R` and put it in your R working directory.

Bivariate Linear Regression in R revisited

Recall from previous class sessions that the main function for carrying out regression in R is `lm()` (short for linear-model).

Read and work through the exercises in Verzani, section 13 (starting at p. 100 in the PDF), for a refresher on bivariate regression. As you work through the material you will notice that Verzani uses a number of functions that are defined in the code file `simple.R`, for example `simple.lm()`. Most of these are what me might call "wrapper functions" — they wrap pre-existing R functions to make them more convenient to use, for example automatically creating useful plots.

> **Assignment 1**: Open up `simple.R` in your text editor and read through the code for `simple.lm()`. Write a description of what each of the major steps in the `simple.lm()` accomplishes. If you want you can embed little snippets of code in your explanation by surrounding the code with `\begin{verbatim}...\end{verbatim}`:

```
\begin{verbatim}
x < - 1:10
y <- 10:20
z <- x + y
\end{!verbatim} # don't include the exclamation mark
```

> These verbatim snippets will not get evaluated when you compile your document to a PDF.

Multiple Regression in R

Like bivariate regression, multiple regression in R is typically conducted using the `lm()` function. Work through Verzani section 14 (p. 109 in the PDF). On p. 114 Verzani demonstrates an application of polynomial regression.

Logistic Regression in R

Logistic regression is a type of 'Generalized Linear Model' (GLM) and hence is fit using the `glm()` function in R. `glm()` can also be used to fit other types of generalized linear models so one must specify the model type as shown below. We will apply logistic regression to study natural selection on a population of house sparrows.

Bumpus' Sparrow Data

Bumpus (1898) described a sample of house sparrows which he collected after a very severe storm. The sample included 136 birds, sixty four of which perished during the storm. Also included in his description were a variety of morphological measurements on the birds and information about their sex and age (for male birds). This data set has become a benchmark in the evolutionary biology literature for demonstrating methods for analyzing natural selection. The bumpus data set is available from the class website as `bumpus-data.txt`.

Bumpus, H. C. 1898. The elimination of the unfit as illustrated by the introduced sparrow, Passer domesticus. (A fourth contribution to the study of variation.) *Biol. Lectures: Woods Hole Marine Biological Laboratory*, 209–225.

Using logistic regression to analyze the Bumpus data

We'll load the Bumpus data set and rename the variable names to shorter, more convenient ones.

```
> bumpus <- read.delim("bumpus-data.txt")
> names(bumpus)
 [1] "line.number"                "sex"                        "age..a...adult..y...young."
 [4] "survived"                   "total.length..mm."          "alar.extent..mm."
 [7] "weight..g."                 "length.of.beak.and.head"    "length.of.humerus..in."
[10] "length.of.femur..in."       "length.of.tibiotarsus..in." "width.of.skull..in."
[13] "length.of.sternal.keel..in."
> split.names <- strsplit(names(bumpus), ".", fixed=T)
> split.names[1]
[[1]]
[1] "line"   "number"
> split.names[8]
[[1]]
[1] "length" "of"     "beak"   "and"    "head"
> first <- function(x){x[1]}
> third <- function(x){x[3]}
> new.names1 <- unlist(lapply(g[1:7],first))
> new.names2 <- unlist(lapply(g[8:13],third))
> new.names <- c(new.names1, new.names2)
> new.names
 [1] "line"        "sex"         "age"         "survived"    "total"       "alar"
 [7] "weight"      "beak"        "humerus"     "femur"       "tibiotarsus" "skull"
[13] "sternal"
> names(bumpus) <- new.names
```

Having made the names more convenient we can now proceed to carrying out the logistic regression:

```
> weight <- bumpus$weight
> survived <- bumpus$survived
> plot(survived ~ weight)
> fit <- glm(formula = survived ~ weight, family=binomial(link=logit))
> summary(fit)

Call:
glm(formula = survived ~ weight, family = binomial(link = logit))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6331  -1.1654   0.8579   1.0791   1.4626

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   8.0456     3.2516   2.474   0.0133 *
```

```
weight       -0.3105     0.1272  -2.441   0.0146 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 188.07  on 135  degrees of freedom
Residual deviance: 181.58  on 134  degrees of freedom
AIC: 185.58

Number of Fisher Scoring iterations: 4
> newx <- seq(20,35,0.5)
> p <- predict.glm(fit, newdata=data.frame(weight = newx), type="response")
> lines(newx,p)
```

Assignment 2: Repeat the logistic regression of survival as a function of body weight for: 1)the female birds and 2) the adult male birds. Produce plots illustrating these regressions for each sex.

LOESS Models

LOESS (aka LOWESS; 'Locally weighted scatterplot smoothing') is a modeling technique that fits a curve (or surface) to a set of data using a large number of local regressions. Local weighted regressions are fit at numerous regions across the data range, using a weighting function that drops off as you move away from the center of the fitting region (hence the 'local' aspect). LOESS combines the simplicity of least squares fitting with the flexibility of non-linear techniques and doesn't require the user to specify a functional form ahead of time in order to fit the model. It does however require relatively dense sampling in order to produce robust fits.

Formally, at each point x_i we estimate the regression coefficients $\hat{\beta}_j(x)$ as the values that minimize:

$$\sum_{k=1}^{n} w_k(x_i)(y_k - \beta_0 - \beta_1 x_k - \ldots - \beta_d x_k^2)^2$$

where d is the degree of the polynomial (usually 1 or 2) and w_k is a weight function. The most common choice of weighting function is called the "tri-cube" function as is defined as:

$$w(x) = (1 - |x|^3)^3, \text{for } |x| < 1$$

$$= 0, \text{for } |x| \geq 1$$

Assignment 3: Write an R function that computes the tri-cube function described above. Create a plot of the tri-cube function over the range (–3,3). Create a second R function that calculates the Gaussian function $f(x) = e^{-x^2}$ and plot that function over the same range in the same plot. If you're struggling with writing the functions you may wish to review Chapter 6 of Paradis, R for Beginners (see wiki) and Chapters 9 (Grouping, loops and conditional execution) and 10 (Writing your own functions) in the "R Introduction" included with R.

The primary parameter that a user must decide on when using LOESS is the size of the neighborhood function to apply (i.e. over what distance should the weight function drop to zero). This is referred to as the "span" in the R documentation, or as the parameter α in many of the papers that discuss LOESS. The appropriate span can be determined by experimentation or, more rigorously by cross-validation.

We'll illustrate fitting a Loess model using data on Barack Obama's approval ratings over the period from November 2008 to September 2010 (`obama-polls.txt` available on the class wiki).

```
> polls <- read.table('obama-polls.txt',header=T,sep="\t")
> names(polls)
[1] "Pollster"   "Dates"      "N.Pop"      "Approve"    "Disapprove" "Undecided"
> dim(polls)
[1] 854   6
# polls in reverse chronological order so let's reverse them
# so we can look at trend from earliest to most recent dates
> approve <- rev(polls$Approve)
> pollnum <- 1:length(approve)
> loess.app <- loess(approve ~ pollnum)
> pred.app <- predict(loess.app, pollnum)

# illustrate Loess with a smaller neighborhood span
> loess.app2 <- loess(approve ~ pollnum, span=0.25)
> pred.app2 <- predict(loess.app2, pollnum)
> lines(pollnum, pred.app2, lwd=2, col='cyan')
```

Take note of how the loess curve changed when we made the span smaller. By decreasing the span we've increased the sensitivity of the model (perhaps overfitting in this case).

> **Assignment 4**: Write an R function that genereates a plot that simultaneously illustrates trends in both approval and disapproval ratings for Barack Obama, showing both the raw data and corresponding loess fits. Use colors and/or shapes to distinguish the two trends. Make sure both your x- and y-axes are scaled to show the full range of the data. Label your axes and create a title in the plot. Aim for a 'publication quality' figure.