Handout #4

Lesson 3: Linear regression

A regression model expresses one variable as a linear combination of one or more other variables. We call the fitted variable the "response," and the other variables are called "predictors." R offers extensive functions for fitting and evaluating linear regression models.

Simple least-squares regression

In simple regression, we have one explanatory variable *Y* and one response variable *X*, and we wish to fit the model $y_i = \beta_o + \beta_1 x_i + \varepsilon_i, \quad i = 1, 2, \ldots, n$.

The least-squares regression model chooses coefficients β_0 and β_1 to minimize the sum of squared differences $\sum_{i=1}^{n} (y_i - f(x_i))^2$, with the additional constraint that the line passes through the sample means ($\bar{x}$, $\bar{y}$). The solution is given by $\hat{y}_i = \bar{y} + r\frac{s_y}{s_x}(x_i - \bar{x})$, where s_x, s_y are the respective sample standard deviations and *r* is the sample correlation coefficient. The errors ε_i are assumed to be independent and normally distributed with mean 0 and variance σ^2.

Preliminary analysis

Before fitting a model, it is important to inspect the data and determine whether or not a linear model seems appropriate. Useful functions are "cor()," "plot()," and "identify()."

cor(x,y): Measure the correlation between two vectors. Also has options to account for missing values.
identify(x,y): Interactively select points on a plot.

Exercise: The dataset "cars" provides 50 observations for the speed of cars and the distances taken to stop (from the 1920s).

Step 1: Univariate analysis.

Load the dataset. Look at the distribution of each variable. How would you describe the distributions? Do they seem normally distributed? Are there any unusual observations?

Step 2: Bivariate analysis.

What is the correlation between "speed" and "dist"? Plot "speed" on the x-axis and "dist" on the y-axis. Use the identify() function to select points. Are there any unusual observations? Does a linear model seem like a good idea?

Step 3: Fitting the model.

You hopefully will recall from your linear models course that the parameter estimates $\hat{\beta}$ are computed using the expression $\hat{\beta} = (X^T X)^{-1} X^T y,$ where X is the matrix of predictor variables and y is the response vector. Since we're fitting a simple regression model with an intercept term, X will look like this:

$$\begin{bmatrix} 1 & x_i \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

Exercise: Construct this matrix for the "cars" dataset, using speed as the predictor variable.

Now that we have the data matrix, the next step is to do the matrix calculations. Here are some helpful functions:

A%*%B: compute the product of two conformable matrices A and B
t(A): take the transpose of matrix A
solve(A): compute the inverse of matrix A (this function can also be used more generally)

Exercise: Use the functions above to compute the parameter estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ for the "cars" dataset. Compute the fitted line and add it to your data plot (use the function "abline" to do this most efficiently). How do the results look?

Now that we've done this the hard way, we'll learn how to let R do the heavy lifting for us…

Fitting models in R

The function "lm( )" is a very general command for fitting least-squares regression models. Syntax:

lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset = NULL, ...)

Formula: takes the form `y~x`, where `y` is the response variable and `x` is the explanatory variable.
Data: the name of the dataset in which the variables are found.

For the "cars" dataset, to fit a model using "speed" as a predictor of "dist", we could type

```
cars.lm=lm(dist~speed, data=cars)
```

This creates a linear model object, with lots of stuff. Type "attributes(cars.lm)" to see what's included.

The summary of a linear models object contains most of the important details.
Type "summary(cars.lm)" to look at the summary output.

Let's look at the fit:

```
plot(cars$speed, cars$dist)
lines(cars$speed, cars.lm$fitted.values)
```

Diagnostic plots: The "plot.lm()" function provides 4 diagnostic plots to evaluate the quality of the fitted model. What are they? What do they measure?

Exercise: Can we do better with this dataset? Should we throw away any observations, or are there transformations that might create a better linear fit? For example, the laws of physics indicate that stopping distance is proportional to the square of speed. Can you adjust your model to account for this relationship? Try it out and look at the results. Does the transformation improve the fit?