Day 3 Exercises without Solutions

Issues with Polynomial Bases

1) This question is about illustrating the problems with polynomial bases. First run

```
set.seed(1)
x <- sort(runif(40)*(10)^.5
y <- sort(runif(40))^0.1
```

to simulate some apparently innocuous data.

(a) Fit 5th and 10th order polynomials to the simulated data using, e.g., `lm(y~poly(x,5))`.

(b) Plot the *x*, *y* data, and overlay the fitted polynomials. (Use the `predict` function to obtain predictions on a fine grid over the range of the x data: only predicting at the data fails to illustrate the polynomial behavior adequately).

(c) One particularly simple basis for a cubic regression spline is $b_1(x) = 1$, $b_2(x) = x$ and $b_{j+2}(x) = |x - x_j^*|^3$ for $j = 1 \ldots q - 2$, where q is the basis dimension, and the x_j^* are knot locations. Use this basis to fit a rank 11 cubic regression spline to the *x*, *y* data (using `lm` and evenly spaced knots).

(d) Overlay the predicted curve according to the spline model, onto the existing *x*, *y* plot, and consider which basis you would rather use.

Orthogonal Matrix Methods

2) Polynomial models of the data from question 1 can also provide an illustration of why orthogonal matrix methods are preferable to fitting models by solution of the 'normal equations' $\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^T\boldsymbol{y}$. The bases produced by `poly` are actually orthogonal polynomial bases, which are a numerically stable way of representing polynomial models, but if a naïve basis is used then a numerically badly behaved model can be produced:

```
form <- paste("I(x^",1:10,")",sep="",collapse="+")
form <- as.formula(paste(form <- paste("y~",form))
```

produces the model formula for a suitably ill-behaved model. Fit this model using `lm`, extract the model matrix from the fitted model object using `model.matrix`, and re-estimate the model parameters by solving the 'normal equations' given above (see `?solve`). Compare the estimated coefficients in both cases, along with the fits. It is also instructive to increase the order of the polynomial by one or two and examine the results (and to decrease it to 5, say, in order to confirm that the QR and normal equations approaches agree if everything is 'well behaved'). Finally, note that the singular value decomposition provides a reliable way of diagnosing the linear dependencies that can cause problems when model fitting. `svd(X)$d` obtains the singular values of the matrix **X**. The largest divided by the smallest gives the 'condition number' of the matrix – a measure of how ill-conditioned computations with the matrix are likely to be.