Part A Simulation and Statistical Programming

Practical 5

Working with Matrices, Solving Equations, Regression

Q1. Using the matrix(), seq() and rep() functions, or cbind() or rbind(), construct the following 4x4 Hankel matrix

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{bmatrix}$$

Use this matrix to create the following matrix

$$B = \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{bmatrix}$$

Q2. (Refer to the reference slides) Calculate the matrix

$$H = C(C^TC)^{-1}C^T,$$

where C is the 4x3 matrix composed of the first three columns of B. Create a diagonal matrix which has the diagonal elements of H as its diagonal entries. Calculate the eigenvalues and eigenvectors of H. Compare the trace of H to the sum of its eigenvalues. Compare the determinant of H with the product of the eigenvalues.

Q3. Let $[x_1, x_2, x_3, x_4, x_5, x_6]^T = [10,11,12,13,14,15]^T$. Find the coefficients of the quintic polynomial f(x) for which $[f(x_1), f(x_2), f(x_3), f(x_4), f(x_5), f(x_6)]^T = [25,16,26,19,21,20]^T$. [Hint : set the problem up as a system of linear equations and use `solve()`].

Q4. The data set `speed.txt` give the times in seconds recorded by the winners in the finals of the men's sprint events (100, 200, 400, 800 and 1500 metres) at each of the 21 Olympic Games from 1900 to 2004 along with the heights above sea level of the different venues.

Fit the following models to the dataset
(i) speed against log(distance)

(ii) speed against distance, distance^2, year, year^2 and log(altitude).

Fit model (i) twice (A) using the `lm()` and `summary()` commands, and (B) using `qr.solve()`.

For each model produce normal qq-plots and comment on these plots.

Q5. Here is an algorithm to solve Ax=b for A an upper triangular matrix, using back substitution.

```
Function: solve Ax=b for x by back-substitution
Input: n x n upper triangular matrix A and n x 1 b
Output: n x 1 vector x solving Ax=b

Step 1: set n equal the number of rows in A
Step 2: If n equals 1 then we are done; return x=b/A.
Step 3: Otherwise (if n>1)

        Step 3.1: create an n x 1 vector x
        Step 3.2: set the last entry of x to be x[n]=b[n]/A[n,n]
        Step 3.3: set b′= b[1:(n-1))] – A[1:(n-1),n]*x[n].
        Step 3.4: set A′=A[1:(n-1),1:(n-1)]

        Step 3.5: solve A′x′=b′ for x′ by back-substitution
        Step 3.6: set x[1:(n-1)]=x′

Step 3.7: return x
```

(a) Implement this algorithm as a recursive function in R. Your function should take as input an upper triangular nxn matrix A and return a solution x satisfying Ax=b.

(b). Create an nxn upper triangular matrix A and an nx1 vector b and check your solution against `backsolve()` and `solve()`.

Q6. Solving Ax=b using solve(A)%*%b is about 4 times slower than solve(A,b). We saw solve(A,b) was about (2/3)n^3 multiplications. Show that solve(A) is about (8/3)n^3 (Hint: if AB=I then the ith column of B (B_i say) is the solution of AB_i=I_i where I_i is the ith column of I; we need the LU factorization of A and then solve AB_i=I_i for i=1,2,…,n).