Statistical Programming: Worksheet 5

1. Polynomials. Find the coefficients of the quintic polynomial $f(x)$ for which

$$(f(2), f(3), f(4), f(5), f(6), f(7))^T = (3, 2, 1, 6, 95, 538)^T$$

[Hint: set the problem up as a system of linear equations and use solve()].

2. Linear Models. The data in `speed.txt` give the times in seconds recorded by the winners in the finals of various men's running events (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.

Read in the data and take a look at it. Calculate the average speed in metres per second of the winner of each race, and add this to the dataframe.

Fit the following models to the dataset

(i) speed against $\log_2$(distance).

(ii) speed against $\log_2$(distance), year and $\log_2$(altitude).

Fit model (i) twice using (a) the `lm()` commands, and (b) using `qr.solve()`.

3. Back Solving. Here is an algorithm to solve $Ax = b$ where A is an upper triangular matrix, using back substitution.

Function: solve $Ax = b$ for x by back-substitution
Input: non-singular $n \times n$ upper triangular matrix A and vector b of length n
Output: vector x such that $Ax = b$

```
1. set $n$ equal the number of rows in $A$;
2. if $n = 1$ return $x = b/A$;
3. otherwise create a vector $x$ of length $n$;
4. set $x_n = b_n/A_{nn}$;
5. set $b' = b_{1:(n-1)} - A_{[1:(n-1),n]}x_n$;
6. set $A' = A_{[1:(n-1),1:(n-1)]}$;
7. solve $A'x' = b'$ for $x'$ by back-substitution;
8. set $x_{[1:(n-1)]} = x'$;
9. return $x$.
```

(a) Implement this algorithm as a recursive function in R. Your function should take as input an upper triangular $n \times n$ matrix A and a vector b of length n, and return a solution x satisfying $Ax = b$.

(b) Create an $n \times n$ upper triangular matrix A and a vector b of length n and check your solution against `backsolve()` and `solve()`.

4. Gram-Schmidt*. One method for obtaining the QR decomposition of an $n \times p$ matrix A is to use the Gram-Schmidt process. The algorithm

```
**Input**  : $n \times p$ matrix $A$.
**Output**: Orthogonal $n \times p$ matrix $Q$ and upper triangular
            $p \times p$ matrix $R$ such that $A = QR$.
for $k \in \{1, \dots, p\}$ do
    Set $r_{kk} = \sqrt{\sum_j a_{jk}^2}$;
    Set $q_{[1:n,k]} = A_{[1:n,k]}/r_{kk}$;
    for $i \in \{k+1, \dots, p\}$ do
        $r_{ki} = \sum_{j=1}^{n} a_{ji} q_{jk}$;
        $a_{[1:n,i]} = a_{[1:n,i]} - r_{ki} q_{[1:n,k]}$;
    end
end
```

(a) Implement the algorithm as a function in R.

(b) Combine your answer to (a) and Question 3 to construct a function for solving linear models from scratch. Apply the function to the data in question 2.

(c) What is the complexity of this algorithm?