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()].

```
> sq <- 2:7
> A <- cbind(sq^5, sq^4, sq^3, sq^2, sq, 1)
> b <- c(3, 2, 1, 6, 95, 538)
> solve(A, b)


                        sq
   1  -17  114 -376  607 -379
```

so the polynomial is $f(x) = x^5 - 17x^4 + 114x^3 - 376x^2 + 607x - 379$.

2. **Linear Models.** The data in `speed.txt` (available on the course website) 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.

```
> dat <- read.table("http://www.stats.ox.ac.uk/~nicholls/PartASSP/speed.txt",
+                        header=TRUE)
> dat$Speed <- 100*dat$Distance.100/dat$Time
```

   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()` command, and (b) using `qr.solve()`.

   For (i)

```
lm(Speed ~ log2(Distance.100), data = dat)



Call:
lm(formula = Speed ~ log2(Distance.100), data = dat)

Coefficients:
       (Intercept)  log2(Distance.100)
            10.719              -1.058


X <- cbind(1, log2(dat$Distance.100))
qr.solve(X, dat$Speed)  # gives the same answer

[1] 10.719340 -1.058174
```

and we see the two methods give the same answer. Each doubling of the length of the race results in a decrease of average winning time of 1.06 ms^{-1}.

For the second model,

```
lm(Speed ~ log2(Distance.100) + Year + log2(Altitude), data = dat)



Call:
lm(formula = Speed ~ log2(Distance.100) + Year + log2(Altitude),
    data = dat)

Coefficients:
       (Intercept)  log2(Distance.100)                Year
          -9.70260            -1.05817             0.01040
    log2(Altitude)
           0.01976
```

There is a steady increase in speed over time, equivalent to about 0.01 ms^{-1} *per year. Higher altitude results in a small increase in speed, equivalent to 0.02 ms*$^{-1}$ *per doubling of the altitude.*

3. **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 for this is below.

```
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.

```
gs <- function(A) {
    n <- nrow(A)
    p <- ncol(A)

    Q <- matrix(0, n, p)
    R <- matrix(0, p, p)

    for (k in 1:p) {
        R[k, k] = sqrt(sum(A[, k]^2))
        Q[, k] = A[, k]/R[k, k]
        for (i in seq(k + 1, length.out = p - k)) {
            R[k, i] = sum(A[, i] * Q[, k])
            A[, i] = A[, i] - R[k, i] * Q[, k]
        }
    }
    return(list(Q = Q, R = R))
}
```

(b) Combine your answer to (a) and to Question 3 from Sheet 4 (recall it implemented a back-solving method) to construct a function for solving linear models from scratch. Apply the function to the data in question 2.

```
## Solution from Problem Sheet 4, q3.
backSolve <- function(A, b) {
    n <- length(b)
    if (nrow(A) != ncol(A))
        stop("A must be a square matrix")
    if (nrow(A) != n)
        stop("Dimensions of A and b must match")

    x = b[n]/A[n, n]
    if (n == 1)
        return(x)
    A2 <- A[-n, -n, drop = FALSE]
    b2 <- b[-n] - A[-n, n] * x

    x = c(backSolve(A2, b2), x)

    return(x)
}

solveLM <- function(X, Y) {
    qr <- gs(X)
    rhs <- t(qr$Q) %*% Y
    backSolve(qr$R, rhs)
}

solveLM(X, dat$Speed)

[1] 10.719340 -1.058174

qr.solve(X, dat$Speed)  # R's version

[1] 10.719340 -1.058174
```

(c) What is the complexity of this algorithm?

The code inside the inner loop is called $p(p-1)/2$ times, and involves approximately $4n$ operations, so the complexity is $2np(p-1) = O(p^2n)$.