Simulation and Statistical Programming: Examples Sheet 4

Problem Sheet 4, due Tuesday 10am Week 8. Please email solutions in a single well-commented R-script to andreas.anastasiou@jesus.ox.ac.uk

1. Let $X$ be an $n \times p$ matrix of fixed covariates with $n > p$, and suppose that $X$ has full column rank $p$.

(a) Explain why the $p \times p$ matrix $X^T X$ is invertible. 
There are various ways to see this but, for example, consider an $n \times n$ matrix $Z$ whose first $p$ columns are $X$, and the rest are orthogonal to each other and to $X$. The matrix has full rank $n$ by construction, so it is invertible, and therefore so is $Z^T Z$. However, the first $p$ columns of $Z^T Z$ are just $X^T X$ with block matrix of 0s below. Hence $X^T X$ has rank $p$.

Consider the linear model given by

$$Y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i,$$

where $\varepsilon_i \sim N(0, \sigma^2)$.

(b) Write down the distribution of $Y_i$, and use it to write out the log-likelihood for $\beta = (\beta_1, \ldots, \beta_p)$.

We have $Y_i \sim N(\beta_1 x_{i1} + \cdots + \beta_p x_{ip}, \sigma^2)$, which gives the log-likelihood

$$l(\beta, \sigma^2) = -\frac{n}{2} \log(2\pi \sigma^2) - \frac{1}{2\sigma^2} \sum_i (Y_i - \beta_1 x_{i1} - \cdots - \beta_p x_{ip})^2$$

(c) Show that the MLE is equivalent to minimising the sum of squares:

$$R(\beta) = \sum_{i=1}^{n} (Y_i - \beta_1 x_{i1} - \cdots - \beta_p x_{ip})^2.$$ 

For any fixed $\sigma^2$, the maximum of $l(\beta, \sigma^2)$ is clearly achieved at the $\beta$ that minimises $R$.

(d) By differentiating and writing the problem as a system of linear equations, show that the MLE is $\hat{\beta} = (X^T X)^{-1} X^T Y$.

Differentiating $R$ with respect to $\beta_j$ gives

$$2 \sum_{i=1}^{n} x_{ij}(Y_i - \beta_1 x_{i1} - \cdots - \beta_p x_{ip}) = 0,$$

which if written as a series of equations for $j = 1, \ldots, p$ gives

$$2X^T Y - 2X^T X \hat{\beta} = 0,$$

and hence the solution is as suggested using the fact that, from (a), $X^T X$ is invertible.

2. Consider the linear model $Y = X\beta + \epsilon$ where $Y$ is a vector of $n$ observations, $X$ is an $n \times p$ matrix with each column containing a different explanatory variable and $\epsilon$ is a vector of $n$ independent normal random errors with mean zero and unknown variance $\sigma^2$. The maximum likelihood estimator for $\beta$ is

$$\hat{\beta} = (X^T X)^{-1} X^T Y.$$ 

The sample variance is

$$s^2 = \frac{1}{n-p} \| X \hat{\beta} - Y \|^2$$

where $p$ is the length of $\beta$. The standard error for $\beta$ is

$$se(\hat{\beta}_i) = s \sqrt{[(X^T X)^{-1}]_{ii}}$$
(a) The trees data give Girth, Height and Volume measurements for 31 trees. Fit the model

\[ Y_i = \beta_1 + x_i^{\text{height}} \beta_2 + x_i^{\text{girth}} \beta_3 + \epsilon_i \]

using the R commands

```r
data(trees)
summary(lm(Volume ~ Girth + Height, data = trees))
```

Call:
lm(formula = Volume ~ Girth + Height, data = trees)

Residuals:
  Min     1Q   Median     3Q    Max
-6.4065 -2.6493 -0.2876  2.2003  8.4847

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -57.9877    8.6382  -6.713 2.75e-07 ***
Girth          4.7082    0.2643  17.816  < 2e-16 ***
Height         0.3393    0.1302   2.607  0.0145  *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.882 on 28 degrees of freedom
Multiple R-squared:  0.948, Adjusted R-squared:  0.9442
F-statistic: 255 on 2 and 28 DF,  p-value: < 2.2e-16

and briefly interpret the output.

(b) Write a function of your own (using `solve()` or your solution to Q3, not `lm()`) to fit a linear model. Your function should take the length 31 vector `trees$Volume` and the 31 x 3 matrix `X = cbind(1, trees$Girth, trees$Height)` as input and return estimates of \( \beta \), the residual standard error \( s \), and the standard errors of each \( \beta_i \). Check your output against the corresponding results from the `summary(lm())` output in (a).

```r
fitLM <- function(X, Y) {
  n <- nrow(X)
p <- ncol(X)

  ## get MLE of beta
  XXi <- solve(t(X) %*% X)
  beta <- XXi %*% (t(X) %*% Y)

  ## get s
  s2 <- sum((t(X) %*% beta - Y)^2)/(n - p)
  ses <- sqrt(s2 * diag(XXi))

  return(list(beta = beta, s2 = s2, ses = ses))
}
```

3. Here is an algorithm to compute the QR factorisation of an \( n \times p \) matrix \( A \) with \( p \leq n \). That
is, it returns an $n \times p$ orthogonal matrix $Q$ and a $p \times p$ upper triangular matrix $R$ such that $A = QR$.

Let $|v|$ denote the Euclidean norm of a vector $v$. Let $A_{[a:b]}$ denote the matrix formed from the columns $a, a+1, \ldots, b$ of $A$.

1. Create $n \times p$ matrix $Q$ and $p \times p$ matrix $R$.
3. If $p = 1$ then we are done; return $Q$ and $R$.
4. Otherwise (i.e. if $p > 1$), set $R_{1[2:p]} = Q_{1[1]}^T A_{[2:p]}$ and $R_{2:p,1} = 0$.
   [Notice that $Q_{[1]} R_{[1:2:p]}$ is an outer product of an $n$ component column vector and a $(p-1)$ component row vector, so $A'$ is a new $n \times (p-1)$ matrix. Either make use the outer() command or, if you use [ be careful to use the drop argument when forming these submatrices.]
6. Compute the QR factorisation of $A'$ (so $A' = Q'R'$ say).
7. Set $Q_{[2:p]} = Q'$ and $R_{[2:p,2:p]} = R'$ and return $Q$ and $R$.

(a) Implement this algorithm as a recursive function in R. Your function should take as input an $n \times p$ matrix $A$ and return two matrices $Q$ and $R$ as a list. State briefly how you checked your function was correct.

```r
my_qr <- function(A) {
  # initialise matrices
  n <- nrow(A)
  p <- ncol(A)
  Q <- matrix(0, n, p)
  R <- matrix(0, p, p)

  # if p=0 then finish
  if (p == 0)
    return(list(Q = Q, R = R))

  # set first columns by renormalising
  R[1, 1] <- sqrt(sum(A[, 1]^2))
  Q[, 1] <- A[, 1]/R[1, 1]

  # now, if n=1 then return this answer,
  if (p == 1)
    return(list(Q = Q, R = R))

  # fill in rest of top row of R
  R[1, -1] <- t(Q[, 1]) %*% A[, -1, drop = FALSE]

  # recurse on Ap
  Ap <- A[, -1] - Q[, 1, drop = FALSE] %*% R[1, -1, drop = FALSE]
  tmp <- Recall(Ap)

  # fill in rest of Q and R
  Q[, -1] <- tmp$Q
  R[-1, -1] <- tmp$R

  return(list(Q = Q, R = R))
}
```
list(Q = Q, R = R)

You should test your function on a few matrices of differing \( n \) and \( p \): check that \( X = QR \) (to numerical precision, you need to ignore small errors) and that \( Q \) is orthogonal.

(b) Using your QR function, and the R command \texttt{backsolve()}, give a least squares solution to the over-determined system

\[
X\beta = Y
\]

where \( X \) and \( Y \) take their values from the \texttt{trees} data in question 2.

\begin{verbatim}
X <- cbind(1, trees$Girth, trees$Height)
Y <- trees$Volume
QR <- my_qr(X)
backsolve(QR$R, t(QR$Q) %*% Y)

[,1]
[1,] -57.9876589
[2,]  4.7081605
[3,]  0.3392512
\end{verbatim}

And we can verify that we get the same answer using R’s built in methods:

\begin{verbatim}
qr.solve(X, Y)

[1] -57.9876589  4.7081605  0.3392512
\end{verbatim}