Problem 1: A "*matrix norm*" is a way of assigning a numerical measurement to a matrix. If $A \in \mathbb{R}^{n\times m}$ is a matrix the four commonly used norms are denoted by,

$$||A||_1, ||A||_\infty, ||A||_2, ||A||_F$$

There are different types of matrix norms, each useful in their own context.

Let $A \in \mathbb{R}^{n\times m}$. The "*Frobenius norm*" is defined as follows,

$$||A||_F \overset{\text{(def)}}{=} \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}(A_{i,j})^2}$$

(square root of the sum of squares, which is how you compute the norm of a vector in multivariable calculus). Sometimes the Frobenius norm is also called the Euclidean norm and can also be denoted by $||A||_E$.

The "*1-norm*" is defined as,

$$||A||_1 \overset{\text{(def)}}{=} \max_{1\le j\le m}\left(\sum_{i=1}^{n}|A_{i,j}|\right)$$

(in other words, add up all the columns in absolute value, and out of all those column sums, pick the largest number).

The "∞-*norm*" is defined as,

$$||A||_\infty \overset{\text{(def)}}{=} \max_{1\le i\le n}\left(\sum_{j=1}^{m}|A_{i,j}|\right)$$

(in other words, add up all the numbers in the rows with absolute value, and out of all those row sums, pick the largest number).

Finally, the 2-norm is the most difficult to define,

$$||A||_2 = \max(\sigma_1, \sigma_2, ..., \sigma_m)$$

where the σ_k are the "*singular values*" of A, i.e.

$$\sigma_k = \sqrt{\lambda_k(A^t A)}$$

Simply put, calculate the eigenvalues of $A^t A$, the largest of the radicals of the eigenvalues is the 2-norm of the matrix.

Create a code in R called:

```
mat.norm(A,type=c(``one'',``inf'',``F'',``2''))
```

which calculates the matrix norm depending on which type of norm you want to use.

Problem 2: Another method for approximating roots is called *Newton's method*. Suppose $f()$ is some function for which we wish to find a root. Let $g()$ denote the derivative of $f()$. Newton's method proceeds by first making an initial guess x_0 (a "*guess*" is just an arbitrary starting value). We then construct the sequence,

$$(x_0, x_1, x_2, ..., x_n)$$

defined by the rule that,

$$x_i = x_{i-1} - \frac{f(x_{i-1})}{g(x_{i-1})}$$

The last number in this sequence, x_n, will be a good approximation for the root.

Write a code in R called `newtons.method(f,g,x0,n)` which calculates the last term in the sequence of approximations.