Solving systems of linear equations with R

The goal of this exercise is to show you how to use R to solve systems of linear equations, and introduce you to the matlib package.

Start RStudio, and use File -> Open -> New R script (Ctrl-Shift-N) for an empty R script window. Type the lines below, and Run them, observing the output.

matlib is a small library of useful R functions for matrix operations. You can load this as follows.

```
library(matlib)
```

1. Create the matrix **A** and the vector **b** to represent two equations in two unknowns, of the form, **A x** = **b**. `showEqn()` shows what they look like as equations, and `plotEqn()` plots them as lines. `Solve()` shows the solution.

   ```
   A <- matrix(c(1, -1, 2, 2), 2, 2, byrow=TRUE)
   b <- c(2, 1)
   showEqn(A, b)
   plotEqn(A, b)
   Solve(A, b)
   ```

   Verify that you understand why the result of `Solve()` given is the solution.

2. The following statements define a set of 3 equations in 3 unknowns, of the form **A x** = **b**. `showEqn()` shows what they look like as equations.

   ```
   A <- matrix(c(7,  5, -3,
                 3, -5,  2,
                 5,  3, -7), 3, 3, byrow=TRUE)

   b <- c(16, -8, 0)
   showEqn(A, b)
   ```

3. The function `R()` finds the rank of a matrix. As we will see in the lecture, a system of equations is *consistent* (have a solution) if r(**A**) = r(**A** | **b**). Note that `cbind()` is used to join matrices and vectors by columns in R.

   ```
   R(A)
   Ab <- cbind(A, b)
   R(Ab)
   ```

4. For consistent equations, the solution is $\mathbf{x} = \mathbf{A}^{-1}\,\mathbf{b}$, so the easy ways to find the solution are to use the `inv()` function or the `Solve()` function. (The basic R function is `solve()`; our `Solve()` function is mainly for tutorial purposes.)

   ```
   # solve for x
   Solve(A, b)
   inv(A) %*% b
   ```

5. Each equation in 3 unknowns corresponds to a plane. The function `plotEqn3d()` shows the planes and the solution. Try it.

   ```
   plotEqn3d(A, b)
   ```

6. Another method is to use the function `echelon()` to reduce the matrix ($\mathbf{A} \mid \mathbf{b}$) to echelon form, which gives ($\mathbf{I} \mid \mathbf{A}^{-1}\mathbf{b}$). The function has some options to show the steps used in finding the solution.

```
echelon(A, b)
echelon(A, b, verbose=TRUE, fractions=TRUE)
```

This method also works for inconsistent equations (where `inv(A)` and `solve(A,b)` give error messages), and shows which equations are inconsistent.

7. All of these methods essentially use elementary row operations (EROs), corresponding to adding multiples of one equation to another (`rowadd`), multiplying equations by constants (`rowmult`) and interchanging equations (`rowswap`) to transform ($\mathbf{A} \mid \mathbf{b}$) to ($\mathbf{I} \mid \mathbf{A}^{-1}\mathbf{b}$). The following demonstrates the first few steps

```
# using row operations to reduce below diagonal to 0
Ab <- cbind(A, b)
(Ab <- rowadd(Ab, 1, 2, 3/2))   # row 2 = row 2 + 3/2 row 1
(Ab <- rowadd(Ab, 1, 3, 1))     # row 3 = row 3 + 1 row 1
(Ab <- rowadd(Ab, 2, 3, -4))
# multiply to make diagonals = 1
(Ab <- rowmult(Ab, 1:3, c(1/2, 2, -1))
# The matrix is now in triangular form
# Could continue to reduce above diagonal to zero
```

8. Using what you've learned so far, setup and find solution(s) to the following equations. Try `inv()`, `Solve()` and `echelon()`.

$$
\begin{aligned}
4x_1 - 1x_2 + 3x_3 &= 2 \\
3x_1 + 5x_2 + 6x_3 &= 4 \\
1x_1 + 2x_2 + 3x_3 &= 5
\end{aligned}
$$