8 Strings and Matrices

8.1 Chars and Strings

We have already seen strings stored as arrays of char. There is also an inbuilt data-type called string that uses double-quotes. A list of strings can be stored in a string array. It can be created using

myStrings = [ "Wayne", "Winifred", "Winston" ];

An individual string can be accessed as in a vector; for example, myStrings(2) will report Winifred. There are many inbuilt string functions. The blanks command will create a blank char-vector of specified length.

8.2 Matrices and tables

A matrix (or table) is like a vector except two-dimensional. For example, the code

M=[ 8 1 6; 3 5 7; 4 9 2 ] produces a matrix that has three rows and three columns, with the semi-colon indicating the end of each row.

One can access individual entries in the table by specifying the row and column; for example M(2,3) is 7. One can extract an entire row or entire column by using a colon: for example, M(2,:) yields the second row, and M(:,3) yields the third column.

The size function returns the dimensions. There are standard linear algebra functions such as transpose and multiplication. There are inbuilt matrices such as eye (the identity), zeros (the all-zero matrix), and ones (the all-one matrix).
A Caesar cypher replaces every letter by a letter later on with wrap around. That is, 'A' becomes 'B', 'B' becomes 'C', and so on, until 'Z' becomes 'A'.

% applies caesar to lower-case string stored as char vector
% goddard 2020
function str = caesar ( str )
    for i=1:length(str)
        if str(i)=='z'
            str(i)='a';
        elseif str(i)>='a' && str(i)<'z'
            str(i)=str(i)+1;
        end
    end
end