Introduction to Modelling

4. Function Examples

Dr. Jim Duggan,
School of Engineering & Informatics
National University of Ireland Galway.

https://github.com/JimDuggan/MATLAB

General form of a function

- A function M-file *name*.m has the following general form

```
function[outarg1, outarg2, …] name(inarg1,...)
% comments to be displayed with help
...
outarg1 = ...;
outarg2 = ...;
...
```

Challenge 4.1

- Write a function (swap) that takes:
  - A two dimensional array (m)
  - A target value (t)
  - A replacement value (r), and
- Replaces all occurrences of t in m with the value r

Challenge 4.2

Write a function **evens(v)** which returns the even values of a vector. A sample test run of the function is shown below.

```
>> v

v =

    3    6    5    3    2    3    1    2    5    1

>> v1 = evens(v)

v1 =

    6    2    2
```

Challenge 4.3

Write a function (m file) that processes elements of a 2-dimensional array on a row-by-row basis. The function should return 2 column vectors, the first containing the minimum value for each row, the second containing the maximum value of each row.

Furthermore, min and max *subfunctions* should be written to calculate the min and max of an individual row (i.e. the MATLAB min and max cannot be used).

Sample data for the problem (1 input and 2 outputs) is shown below.

$$Input = \begin{pmatrix} 10 & 20 \\ 50 & 40 \\ 80 & 60 \end{pmatrix} \quad Min = \begin{pmatrix} 10 \\ 40 \\ 60 \end{pmatrix} \quad Max = \begin{pmatrix} 20 \\ 50 \\ 80 \end{pmatrix}$$

Challenge 4.4

- Explain what is happening in the following four lines of MATLAB code, and show what the values of y1 and y2 will be.

```
f1 = @max
f2 = @min
y1 = feval(f1,10,20);
y2 = feval(f2, 10, 20);
```

- What are the potential benefits of using @ and feval, and name a MATLAB function that makes use of these mechanisms.

Challenge 4.5

Write a function (m file) that takes a 2-dimensional array and an input number. It should then create an output 2-dimensional array that contains only those values of the 2-dimensional array that are greater than the input number. For example, if input number is 5, and the input array (A) is

```
A =
   1   2   3
   4   5   6
   7   8   9
```

Then the function output should be.

```
ans =
   0   0   0
   0   5   6
   7   8   9
```

Challenge 4.6

- Explain what is happening in the following MATLAB code, and determine the values (and type) of the output.

```
f = @(x) [sum(x); min(x); max(x)]

f(1:5)
```

Challenge 4.7 - Imputation

- In statistics, **imputation** is the process of replacing missing data with substituted values.
- **Mean imputation**
  - Simply calculate the mean of the observed values for that variable for all individuals who are non-missing.
  - It has the advantage of keeping the same mean and the same sample size, but many, many disadvantages. Pretty much every method listed below is better than mean imputation.

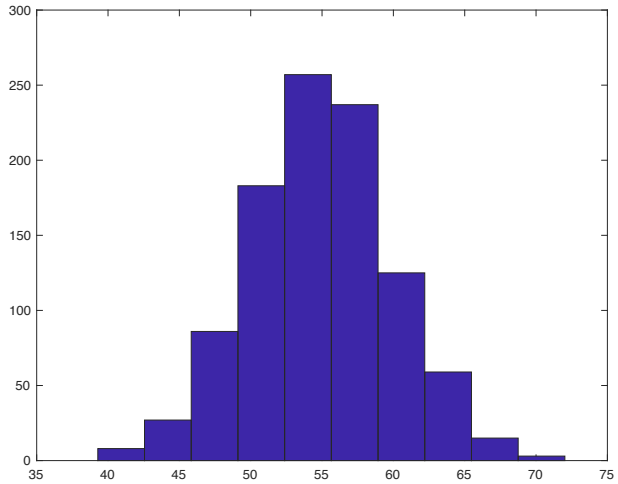
https://www.theanalysisfactor.com/seven-ways-to-make-up-data-common-methods-to-imputing-missing-data/

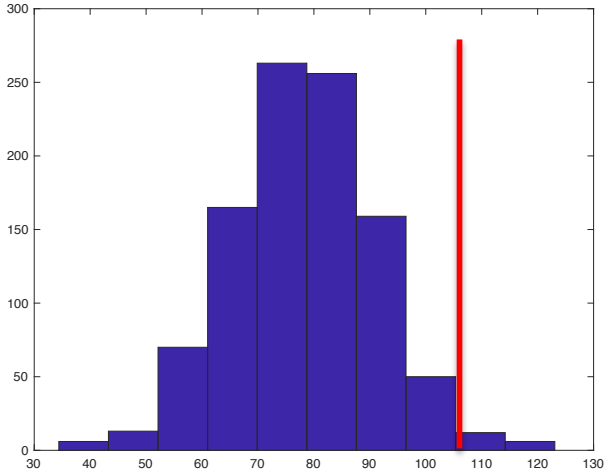
Example

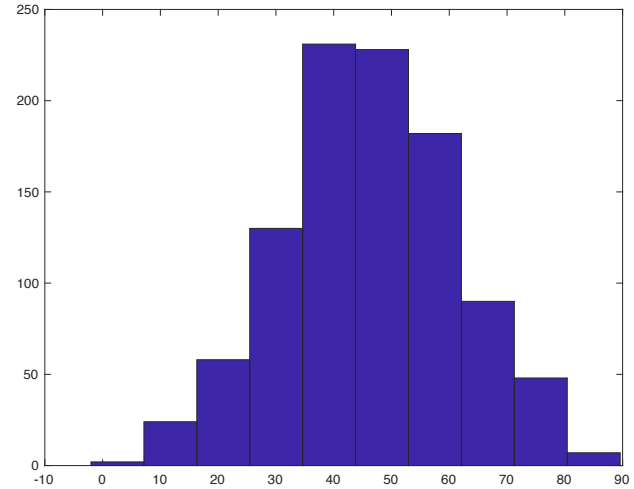
```
clear;
N = 1000;

mean1 = 55;  sd1 = 5;
mean2 = 79;  sd2 = 13;
mean3 = 46;  sd3 = 15;
mean4 = 85;  sd4 = 20;
mean5 = 67;  sd5 = 2;
mean6 = 54;  sd6  = 40;

rng(100);
sub1 =  sd1 .* randn(N,1 ) + mean1;
sub2 =  sd2 .* randn(N,1 ) + mean2;
sub3 =  sd3 .* randn(N,1 ) + mean3;
sub4 =  sd4 .* randn(N,1 ) + mean4;
sub5 =  sd5 .* randn(N,1 ) + mean5;
sub6 =  sd6 .* randn(N,1 ) + mean6;

% (1) create results matrix
res = [sub1 sub2 sub3 sub4 sub5 sub6];
```

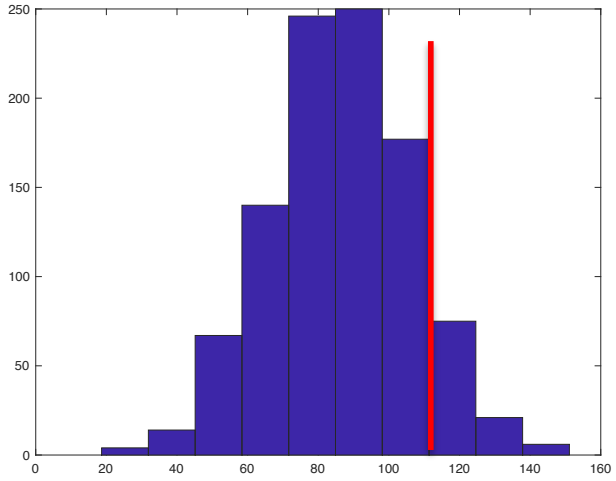
```
>> res(1:10,:)

ans =

   55.8047   74.7572   47.9751  101.5216   64.8236  116.7540
   51.9244   64.3171   45.9122   72.9453   68.2830   58.6920
   53.8050  116.6643   26.6919   71.2220   71.4543   60.4692
   58.0749   68.6322   45.9255   56.0552   67.9900   52.9738
   50.4930   90.7065   41.0240   83.2052   64.8992   55.0326
   56.7404   76.8842   64.5532   66.0561   68.1651   77.2344
   59.7398   68.7360   55.0958   67.0041   68.8075  115.8837
   46.3506   56.5778   40.7467   76.8292   69.1251  -26.5840
   55.4856   72.0457   48.6464   45.9657   68.2401   47.1588
   61.8929   63.7574   19.7120   65.9481   68.4181   37.0947
```

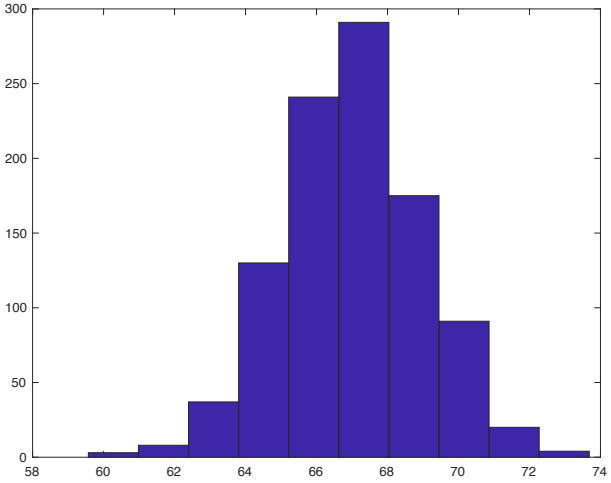
Histograms (explore data) (hist sub$_n$)

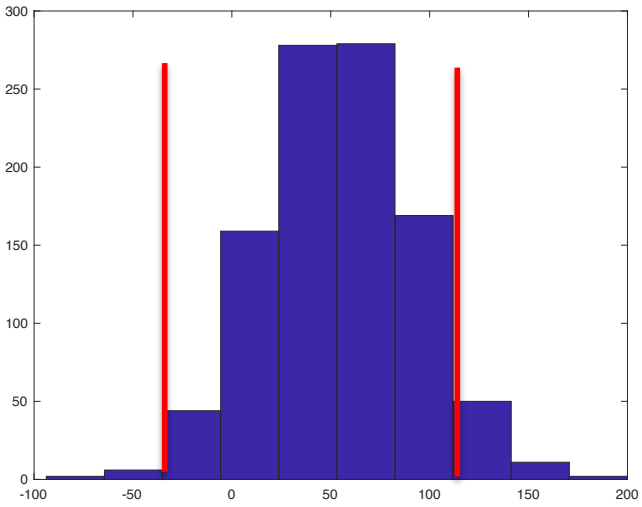












Our process (1/2)

- Write a function called clean
  - Takes an input matrix
  - Takes a validator function (one for > 100, one for < 0)
  - Returns a logical matrix where a 1 means an invalid value
- For the results data set
  - Call clean to find values > 100
  - Call clean to find values < 0
  - Combine matrices to have the location of all invalid values (hint use | operator)

Our process (2/2)

- Write a function called impute
  - Takes an input matrix and the invalid logical matrix information
  - Finds the columns with problematic data (hint use sum with the logical matrix)
  - For each column, replaces the invalid values with the mean of the remaining valid values
  - Returns the imputed matrix.

Results

```
res =

   55.8047   67.0825   70.8806   62.3432   67.2666   50.2265
   51.9244   65.5363   40.1486   78.5100   62.4884   60.1895
   53.8050   76.0861   29.8380   58.2788   66.2023   24.8250
   58.0749  114.1285   42.3620   92.6122   67.3995   26.5756
   50.4930   85.7630   22.3194  125.2700   69.4458   -2.3720
   56.7404   73.0237   32.5792   56.2800   66.3141   16.4034
   59.7398   88.9579   53.7458  105.8411   66.7125  -10.1357
   46.3506   68.5649   21.5531   89.8909   67.8241   54.3835
   55.4856   75.5915   50.4526  112.7119   69.4578   37.5689
   61.8929   99.5852   50.1755   89.9657   69.8418   58.8013

new_res =

   55.8047   67.0825   70.8806   62.3432   67.2666   50.2265
   51.9244   65.5363   40.1486   78.5100   62.4884   60.1895
   53.8050   76.0861   29.8380   58.2788   66.2023   24.8250
   58.0749   77.7990   42.3620   92.6122   67.3995   26.5756
   50.4930   85.7630   22.3194   75.4115   69.4458   41.1217
   56.7404   73.0237   32.5792   56.2800   66.3141   16.4034
   59.7398   88.9579   53.7458   75.4115   66.7125   41.1217
   46.3506   68.5649   21.5531   89.8909   67.8241   54.3835
   55.4856   75.5915   50.4526   75.4115   69.4578   37.5689
   61.8929   99.5852   50.1755   89.9657   69.8418   58.8013
```

```
>> invalid_all

invalid_all =

  10×6 logical array

   0   0   0   0   0   0
   0   0   0   0   0   0
   0   0   0   0   0   0
   0   1   0   0   0   0
   0   0   0   1   0   1
   0   0   0   0   0   0
   0   0   0   1   0   1
   0   0   0   0   0   0
   0   0   0   1   0   0
   0   0   0   0   0   0
```