Computing with MATLAB™

Part 4

Icahn School of Medicine at Mount Sinai

SBCNY

Outline

Scripts versus functions in MATLAB

Local versus global variables

MATLAB scripts versus functions

What if we acquired fluorescence data that looked like this?

sampledata1.DAT

time	Blue	Red	Green
0	2.0348	1.0003	4.9707
0.25	1.9853	1.008	4.9989
0.5	2.0163	0.98505	4.9997
0.75	1.9754	1.0045	5.0003
1	1.9956	1.0035	4.9963
1.25	1.9576	0.98667	5.0036
1.5	1.9918	1.011	5.0304
1.75	1.9785	1.0058	4.9988
2	1.9853	0.9914	5.0004
2.25	1.993	1.044	5.0145
2.5	2.003	1.0132	5.0222

plus hundreds more data points

We’ll write a script to plot these data

MATLAB scripts versus functions

A sample script

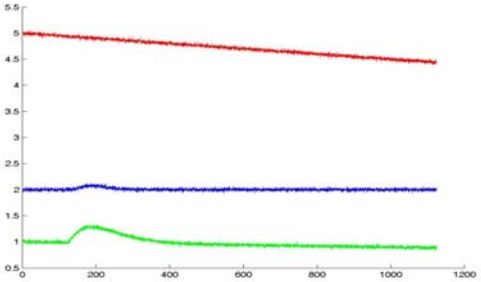
samplescript.m

```
data = dlmread('sampledata1.DAT') ;
time = data(:,1) ;
f1 = data(:,2) ;
f2 = data(:,3) ;
f3 = data(:,4) ;
figure
hold on
plot(time,f1,'b')
plot(time,f2,'g')
plot(time,f3,'r')
```

MATLAB scripts versus functions

Run a script by typing its name

```
>> clear all
>> samplescript
>> whos
  Name      Size      Bytes  Class
  data    4501x4    144032  double array
  f1      4501x1     36008  double array
  f2      4501x1     36008  double array
  f3      4501x1     36008  double array
  time    4501x1     36008  double array
Grand total is 36008 elements using 288064 bytes
```



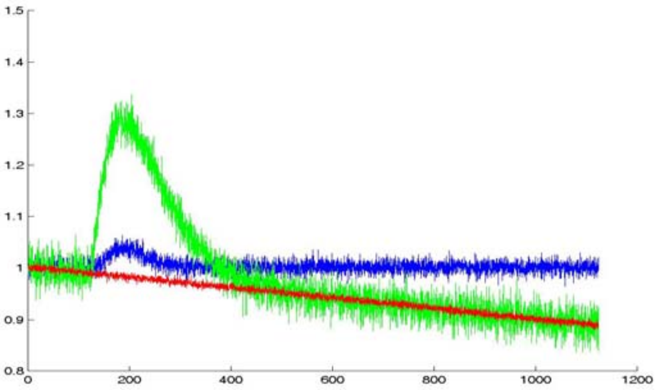
But relative changes are important, so let’s normalize each trace to fluorescence in the first 50 milliseconds.

MATLAB scripts versus functions

To plot normalized traces, modify samplescript.m

```
data = dlmread('sampledata1.DAT') ;
time = data(:,1) ;
f1 = data(:,2) ;
[minimum,index] = min(abs(time-50)) ;
sum = 0 ;
for i=1:index
  sum = sum + f1(i) ;
end
f1avg = sum/index ;
f1_norm = f1/f1avg ;

f2 = data(:,3) ;
sum = 0 ;
for i=1:index
  sum = sum + f2(i) ;
end
f2avg = sum/index ;
f2_norm = f2/f2avg ;

f3 = data(:,4) ;
sum = 0 ;
for i=1:index
  sum = sum + f3(i) ;
end
f3avg = sum/index ;
f3_norm = f3/f3avg ;
```

```
figure
hold on
plot(time,f1_norm,'b')
plot(time,f2_norm,'g')
plot(time,f3_norm,'r')
```



MATLAB scripts versus functions

To simplify, create a function that performs the normalization

samplescript.m

```
data = dlmread('sampledata1.DAT') ;
time = data(:,1) ;
[minimum,index] = min(abs(time-50)) ;
f1 = data(:,2) ;
f1_norm = normalize(f1,index) ;

f2 = data(:,3) ;
f2_norm = normalize(f2,index) ;

f3 = data(:,4) ;
f3_norm = normalize(f3,index) ;

figure
hold on
plot(time,f1_norm,'b')
plot(time,f2_norm,'g')
plot(time,f3_norm,'r')
```

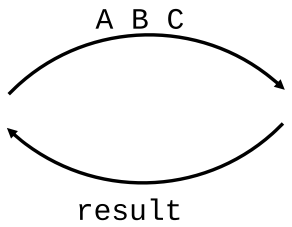
normalize.m

```
function norm = normalize( ...
    vector,numpoints)

sum = 0 ;
for i=1:numpoints
  sum = sum + vector(i) ;
end
average = sum/numpoints ;
norm = vector/average ;

return
```

MATLAB scripts versus functions

Schematic relationship between scripts and functions

`result = samplefunction(A,B,C) ;`



samplefunction.m

```
function a = samplefunction(x,y,z)
```

[A,B,C in script are called x,y,z in function, but this is okay]

```
  LOTS OF IMPORTANT MATLAB CODE

  a = something ;

return
```

[a in function will be called result in script, but this is okay]

After function is called, variables defined within function are gone.

But what if I need to use exactly the same variables in both script & function?
Answer: declare these to be `global` variables

MATLAB functions with global variables

Let’s make `index`, indicating number of points to average, a `global` variable

samplescript.m

```
global index
data = dlmread('sampledata1.DAT') ;
time = data(:,1) ;
[minimum,index] = min(abs(time-50)) ;
f1 = data(:,2) ;
f1_norm = normalize(f1) ;

f2 = data(:,3) ;
f2_norm = normalize(f2) ;

f3 = data(:,4) ;
f3_norm = normalize(f3) ;

figure
hold on
plot(time,f1_norm,'b')
plot(time,f2_norm,'g')
plot(time,f3_norm,'r')
```

normalize.m

```
function norm = normalize(vector)
global index

sum = 0 ;
for i=1:index
  sum = sum + vector(i) ;
end
average = sum/index ;
norm = vector/average ;

return
```

MATLAB functions with global variables

Let’s make `index`, indicating number of points to average, a `global` variable

samplescript.m

```
global index
data = dlmread('sampledata1.DAT') ;
time = data(:,1) ;
[minimum,index] = min(abs(time-50)) ;
f1 = data(:,2) ;
f1_norm = normalize(f1) ;

f2 = data(:,3) ;
f2_norm = normalize(f2) ;

f3 = data(:,4) ;
f3_norm = normalize(f3) ;

figure
hold on
plot(time,f1_norm,'b')
plot(time,f2_norm,'g')
plot(time,f3_norm,'r')
```

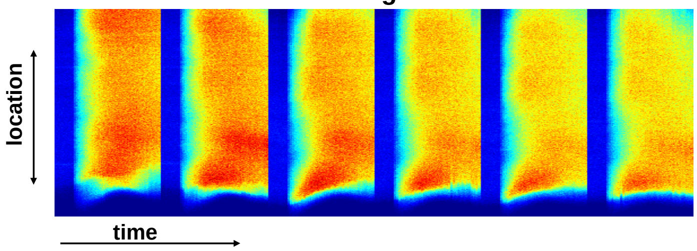
normalize.m

```
function norm = normalize(vector)
global index

sum = 0 ;
for i=1:index
  sum = sum + vector(i) ;
end
average = sum/index ;
norm = vector/average ;

return
```

A practical example of a function

Reading in data from the microscope

Confocal line-scan image of Ca^{2+} in heart cell



The file stored on the computer contains more than the data:

which laser was used?
laser power
what optical filters were used?
microscope objective
scanning speed
when was the recording made?
etc.

Moreover, the file is stored in a proprietary format

A practical example of a function

A MATLAB routine to extract this information from the file

```
filename = 'Image 1.lsm' ;
fileid = fopen(filename,'r') ;

currentoffset = 8 ;
fseek(fileid,currentoffset,-1) ;

  directoryentries = fread(fileid,1,'uint16') ;
  tag = fread(fileid,1,'*uint16') ;
  datatype = fread(fileid,1,'*uint16') ;
  numvalues = fread(fileid,1,'*uint32') ;
  tagvalues = fread(fileid,1,'*uint32') ;
  if (tag == 254 && tagvalues == 0)
    for ii=2:directoryentries
      tag = fread(fileid,1,'uint16') ;
      datatype = fread(fileid,1,'uint16') ;
      numvalues = fread(fileid,1,'uint32') ;
      tagvalues = fread(fileid,1,'uint32') ;
      if (tag == 256)
        xsize = tagvalues ;
      end
      if (tag == 257)
        ysize = tagvalues ;
      end
```

**[These details required to locate the data.
But they are not of interest in general]**

**[To avoid copying and pasting all these commands
each time, save these in a function]**

etc.

This represents roughly 20% of the commands

A practical example of a function

Save the boring commands in a MATLAB function

```
function data = readzeiss(file)

  LOTS OF IMPORTANT MATLAB COMMANDS

  data = something ;

return
```

Save this in the file 'readzeiss.m'

Now, at the MATLAB prompt, type:

```
>> myfile = 'Image 14.lsm' ;
>> mydata = readzeiss(myfile) ;
>> whos mydata
  Name         Size                    Bytes  Class

  mydata     512x20000              20480000  uint16 array

Grand total is 10240000 elements using 20480000 bytes
```

Summary

A *script* is a series of MATLAB commands saved to a file with a .m extension that can be executed by typing the filename at the MATLAB command window.

A *function* must be defined as such. A function may return output and may require certain types of variables as input.

Variables defined within a function are lost once the function is finished.

Variables can be defined as global if they are to be used in both scripts and functions.