1-4

1. Length, Size and Numel in MATLAB

```
length(random_number_matrix)
% important! "length" returns the length of the longest dimension,
% regardless of how many dimensions there are!

% you can use size to find the sizes of all dimensions:
size(twinkies)
size(twinkies,1) % or only specific dimensions...

numel(twinkies) % numel stands for 'total number of elements'
```

Length()只 return 最長的 dim.的長度。
Size()則需另指定輸出長度的 dim.。
Numel()則會 return 所有 element 的數量。

2. Repmat

```
meanOverTimeRepmat = repmat(meanOverTime,1,size(mat,2));
whos mat meanOverTime*
% repmat takes 3 inputs: the matrix you want to replicate, the number of
% times to replicate it over rows, and the number of times to replicate it
% over columns (you can also input more dimensions). We want to replicate
% this matrix only over time points (the size of the second dimension of
% mat). Now the subtraction works:

matmean = mat - meanOverTimeRepmat;
```

```
A = diag([100 200 300])
```

A = 3×3

100	0	0
0	200	0
0	0	300

```
B = repmat(A,2,3)
```

B = 6×9

100	0	0	100	0	0	100	0	0
0	200	0	0	200	0	0	200	0
0	0	300	0	0	300	0	0	300
100	0	0	100	0	0	100	0	0
0	200	0	0	200	0	0	200	0
0	0	300	0	0	300	0	0	300

3. Bsxfun

```
% bsxfun is a useful function for fast and easy array and matrix manipulations.
% It was introduced to Matlab fairly recently, so older versions of Matlab
% do not have this utility.

% for example, the following function will add 4 to a random matrix:
bsxfun(@plus,randn(10),4)

% this might not seem any better than "randn(10)+4" and for this small
% case, it isn't. bsxfun is more useful because it performs
% singleton-expansion, which means you may be able to avoid using repmat.
```

```
% For example, imagine a dataset with 100 channels and 100,000 time points:
a  = rand(100,100000);

% To subtract the mean of the entire time series:
am = a - repmat(mean(a,2),1,size(a,2));
```

```
% The previous line crashes because the sizes of a and its mean are not the
% same. However, bsxfun expands this automatically
am = bsxfun(@minus,a,mean(a,2));
```

C = bsxfun(fun,A,B)
快速計算 matrix 的運算。

4. Plot3

```
% You might instead have a 3D matrix, e.g.,
data3d = randn(3,30);
plot3(data3d)
% Although the previous line seems like it should work, it unfortunately
% doesn't. You'll need to input each dimension separately:
plot3(data3d(1,:),data3d(2,:),data3d(3,:),'ko-','linew',3,'markerface','m')
% you can use the same extra inputs to define line features as you would with the normal plot function
axis off
axis square % also try tight and normal
```

需要指定 dims.

5. Set(gac,)

```
plot(1:10,rand(10,3))
set(gca,'xtick',1:2:9); % gca = "get current axis"; note the parameter-value pair afterwards
set(gca,'xtick',1:2:9,'xticklabel',{'one';'three';'five';'seven';'nine'})
% can put multiple parameter-value pairs in one function

% the complement to set is get. type "get(gca)" to see a list of parameters
% you can change
get(gca)
% you can also access (and return output from) axis properties:
axis_ylim = get(gca,'YLim'); % axis_ylim is the lower and upper bounds of the y-axis

% you can also assign axis properties using variables or functions:
the_ylim_i_want = [-.3 -cos(pi)];
set(gca,'YLim',the_ylim_i_want);
```

使用 gac 更改 axis 的訊息。

6. Meshgrid

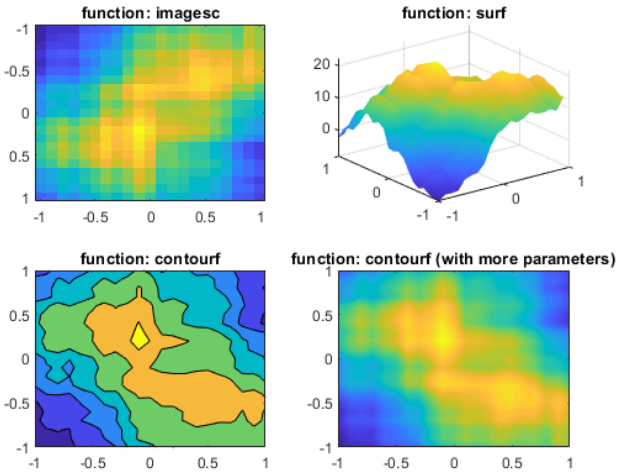
```
x = 1:3;
y = 1:5;
[X,Y] = meshgrid(x,y)
```

```
X =                 Y =

     1     2     3     1     1     1
     1     2     3     2     2     2
     1     2     3     3     3     3
     1     2     3     4     4     4
     1     2     3     5     5     5
```

[X,Y] = meshgrid(x,y)
基於向量 x 和 y 中包含的座標返回二維網格(grids)。
X 是一個矩陣，每個 rows 是一個 copy of x；
Y 也是一個矩陣，每個 column 是一個 copy of y。
座標 X 和 Y 表示的網格有 length(y)個 rows 和 length(x)個 column。

7. Image with Scaled Colors




```
% there are other functions you can use for 2D data, including:
figure
data = conv2(gaus2d,randn(100),'same'); % 2D convolution

subplot(221) % that if you don't use variables and have fewer than 10 subplots, commas are not necessary
imagesc(xyrange,xyrange,data)
title('function: imagesc')

subplot(222)
surf(xyrange,xyrange,data)
shading interp
title('function: surf')

subplot(223)
contourf(xyrange,xyrange,data)
title('function: contourf')

subplot(224)
contourf(xyrange,xyrange,data,40,'linecolor','none')
title('function: contourf (with more parameters)')
```

1-5

1. 3 Ways to Find Value Index

```
time2plot = 300; % in ms!

[minval, minidx] = min(abs(EEG.times-time2plot));
%讓timepoint 300作為index，而非第300個timepoint
```

```
freqIwant = 23; % in hz

% use min(abs trick to find closest frequency to 23 Hz
[~,frexidx] = min(abs(frex-freqIwant));
```

```
freqIwant = 23

% the function dsearchn also works
frexidx = dsearchn(frex',freqIwant);
```

2. Way to Find String Index

```
% the electrode label that we want to analyze
electrodeName = 'p1'; % case doesn't matter

% find the channel number that corresponds to this label
electrodeidx = strcmpi(electrodeName,{EEG.chanlocs.labels});
```

1-6

1. Randsample

```
random_trial_to_plot = randsample(EEG.trials,1);
% random sample one num. from EEG.trials
```

2. Squeeze

```
A = zeros(2,1,2);
A(1:2,:,1) = [1 2]';
% Same as A(:,:,1) = [1 2]'
A(1:2,:,2) = [3 4]';
A
```

```
A =
A(:,:,1) =

     1
     2

A(:,:,2) =

     3
     4
```

```
B = squeeze(A)
```

```
B = 2×2
     1     3
     2     4
```

刪除長度為 1 的維度。如上，將 2*1*2 轉為 2*2。

3. ERP

```
% compute ERP
erp = mean(EEG.data(channel_index, :, :), 3);
```

Take mean along with trails.

1-7

1. Linspace

```
% define XY points for interpolation
interp_detail = 100;
interpX = linspace(min(elocsX)-.2,max(elocsX)+.25,interp_detail);
interpY = linspace(min(elocsY),max(elocsY),interp_detail);
```

```
y1 = linspace(-5,5,7)
```

```
y1 = 1×7

   -5.0000 -3.3333 -1.6667 0 1.6667 3.3333 5.0000
```

y = linspace(x1, x2, n)
生成線性且等距向量。從 x1 至 x2，共 n 筆。