MATLAB tips and tricks
• Most of this will be demonstrated in the context of EEG data (plus some simple made-up examples as well)

• Note: Array and vector are used interchangeably (vector/array is a 1xn or nx1 matrix)

• General MATLAB review (matrices, accessing matrices, : for range of values)
‘find’ function

• gives indices for elements in vector (or matrix*) that correspond to some condition

• CAUTION: hard to use with matrices (works best on vectors)
‘find’ function: simple example

• Find indices for all vowels in a word

```matlab
>> A = 'hello'
A =
 hello

>> B = find(A=='e' | A=='o' | A=='a' | A=='i' | A=='u')
B =
    2   5

>> A(B)
ans =
eo
```
‘find’ function: practical example

Find the indices for a particular label given labels and features

- Labels: \([n \times 1]\) vector of corresponding classes by row
- Features: \([n \times m]\) matrix of features, where each row corresponds to row in label

```matlab
>> labels
labels =
5
5
5
5
5
1
1
1
1
1
```

```matlab
>> features
features =
1.6305 0.7378 0.9487 1.6514 3.5395
0.5767 0.4743 1.5723 1.3124 2.9345
1.5333 1.9259 2.1895 0.9014 2.6990
1.2775 1.6875 0.7507 0.1554 2.5405
0.6634 2.3156 1.4613 0.8907 2.4459
0.6863 0.8606 0.1970 1.0034 1.0242
0.6653 0.6628 2.0426 2.1752 1.7722
0.5874 1.3714 0.7198 0.7899 2.1320
1.1779 1.7268 0.6552 1.1026 1.9455
0.9677 1.0076 1.1190 0.9875 2.2633
```
‘find’ function: practical example

>> cur_type_indices = find(labels==5)
cur_type_indices =
    1
    2
    3
    4
    5

>> cur_features = features(cur_type_indices, :)
cur_features =

    1.6305   0.7378   0.9487   1.6514   3.5395
    0.5767   0.4743   1.5723   1.3124   2.9345
    1.5333   1.9259   2.1895   0.9014   2.6990
    1.2775   1.6875   0.7507   0.1554   2.5405
    0.6634   2.3156   1.4613   0.8907   2.4459
Cell Arrays

• Allow you to store heterogenous data (different data types, and different size)
Cell Arrays

>> C = {}  
C = 
    {}  
>> C{1} = [1 2 3]  
C =  
    [1x3 double]  
>> C{2} = 'hello'  
C =  
    [1x3 double]    'hello'

>> C{3} = 'goodbye'  
C =  
    [1x3 double]    'hello'    'goodbye'
Cell Arrays: () vs. {} assignment

• When using cell arrays, there is a concept of the ‘cell’ and ‘the data within the cell’

• Almost always want to use {} both for assignment and accessing data
Cell Arrays

• I like to use cell arrays to store strings for conditions

```matlab
>> conditions = {'right MI', 'no MI'}
```
```
conditions =
    'right MI'    'no MI'
```
Struct

• Stores data in fields; can think of each individual field as a variable (stored in a container)

>> A.name = 'Willy';
>> A.age = 23;
>> A
A =
    name: 'Willy'
    age: 23
>> A.name
ans =
Willy
Struct Array

- An array made up of structs
- Access elements same way as arrays using parenthesis
- `name_of_struct(1)`: returns the struct at index 1 in the struct array `name_of_struct`
- `name_of_struct(2).field`: returns whatever data stored in field
Struct Array

• Because it is an array, all arrays have to have the same data type. For struct arrays, this means all fields have to be the same for every struct element
Struct Array

>> A = [] % make sure to clear your struct variable
A =
    []
>> A(1).name = 'Willy'
A =
    name: 'Willy'
>> A(1).age = 23
A =
    name: 'Willy'
    age: 23
>> A(2).name = 'Jack'
A =
    1x2 struct array with fields:
      name
      age
>> A(2)
ans =
    name: 'Jack'
    age: []
Struct Array: Combining data from fields of various struct elements

>> A(2).age = 25
A =
1x2 struct array with fields:
    name
    age
>> A(1)
ans =
    name: 'Willy'
    age: 23
>> A(2)
ans =
    name: 'Jack'
    age: 25
>> {A.name}
ans =
    'Willy'    'Jack'
>> vertcat(A.age)
ans =
    23
    25
How I like to use these techniques

• events: a vector with numbers representing different events during the experiment (11 – trial ended)
• time: a vector with time in samples corresponding to the events vector
• data: a [samples x channels] matrix of EEG data

```matlab
condition_iter = 1;
segmented_data = struct('condition', [], 'data', []);
event_idx = 1;

while events(event_idx)~=0 && event_idx < length(events)
    event_idx = event_idx + 3;
    condition_name = feval(conversion_func, events(event_idx));
    data_start_idx = event_idx + 2;
    data_stop_idx = event_idx + 3;
```
How I like to use these techniques

temp_data = data(time(data_start_idx):time(data_stop_idx), :);
condition_index = find(strcmp({segmented_data.condition}, condition_name));
    if ~isempty(condition_index) % already exists
        segmented_data(condition_index).data = cat(3, segmented_data(condition_index).data, data);
    else % condition doesn't exist yet
        segmented_data(condition_iter).data = data;
        segmented_data(condition_iter).condition = condition_name;
        condition_iter = condition_iter + 1;
    end
event_idx = event_idx + 5;
end
>> segmented_data
segmented_data =
1x2 struct array with fields:
  condition
  data
>> segmented_data(1)
ans =
  condition: 'no MI'
  data: [1537x1x16 double]
>> segmented_data(2)
ans =
  condition: 'right MI'
  data: [1537x1x16 double]
>> all_conditions = {segmented_data.condition}
all_conditions =
  'no MI'  'right MI'
colors = 'brgkbry';
figure; hold on;
for class_iter = 1:numel(segmented_data) % class conditions loop
    channel_data = squeeze(segmented_data(class_iter).data(:,channel_iter,:));
    temp_mean = mean(channel_data, 2);
    temp_std = std(channel_data, 1, 2);
    t = 1/fs:1/fs:length(temp_mean)/fs;

    plot(t, temp_mean, colors(class_iter));
    temp_handle = plot(t, temp_mean-temp_std, [':',colors(class_iter)));
    set(get(get(temp_handle,'Annotation'),'LegendInformation'), 'IconDisplayStyle','off');
    temp_handle = plot(t, temp_mean+temp_std, [':',colors(class_iter)));
    set(get(get(temp_handle,'Annotation'),'LegendInformation'), 'IconDisplayStyle','off');
end
legend({segmented_data.condition});
Plot result
YLIM = [0 4];
conditions = {'12 Hz', '22 Hz'};
f = [12 15 17 20 22];
sample_types = [12 22];

for iter = 1:nnnumel(sample_types)
    cur_type = sample_types(iter);
    cur_type_indices = find(labels==cur_type);
    cur_features = features(cur_type_indices,:);

    cur_mean = mean(cur_features);
    cur_ste = std(cur_features)/sqrt(size(cur_features,1));

    condition_freq_idx = find(f==sample_types(iter));
    figure; hold on;
    errorbar(f, cur_mean, cur_ste, '.b');
    errorbar(f(condition_freq_idx), cur_mean(condition_freq_idx), cur_ste(condition_freq_idx), '.r');
    title(conditions{iter});
ylim(YLIM);
end
Error Bar Results

12 Hz

22 Hz
Plot line colors and displays

- [www.mathworks.com](http://www.mathworks.com)
- Click ‘support’ near top of page
- Click ‘product documentation’, click ‘MATLAB’
- Click ‘index’, under ‘L’, search for ‘LineSpec’
- Will explain different options
Iterating over vectors

• Be careful, iterating over column vectors does not work

>> A = [1 2 3 4];

>> B = A';
Iterating over vectors

- Good (row vector)

```matlab
>> for iter = A
    disp(iter);
    disp('yes');
end
```

1
yes
2
yes
3
yes
4
yes

- Bad (column vector)

```matlab
>> for iter = B
    disp(iter);
    disp('yes');
end
```

```
    1
yes
    2
    3
    4
    yes
```