Matlab lecture 1: Analysing E-prime output with Matlab

Sunghyon Kyeong

Severance Biomedical Science Institute,
Yonsei University College of Medicine

Matlab lecture 1 (October 1, 2016):
Analysing E-prime output with Matlab
about 2 hours

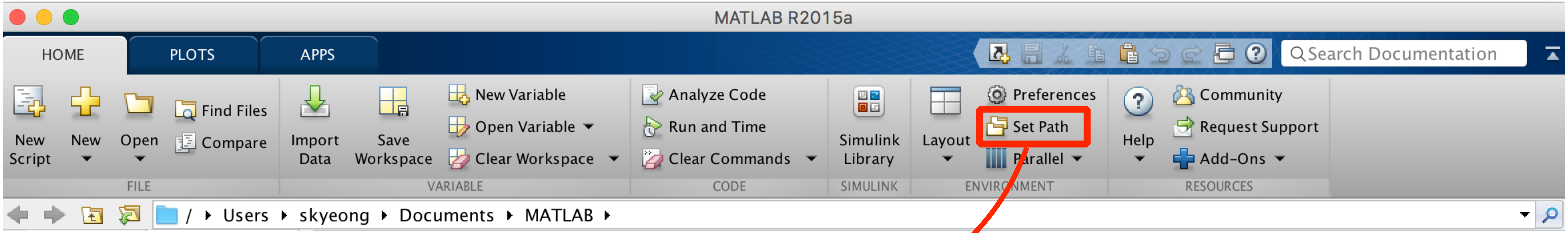
Matlab lecture 2 (Early November, 2016):
Audio-visual stimuli using Psychtoolbox
at least 3 hours

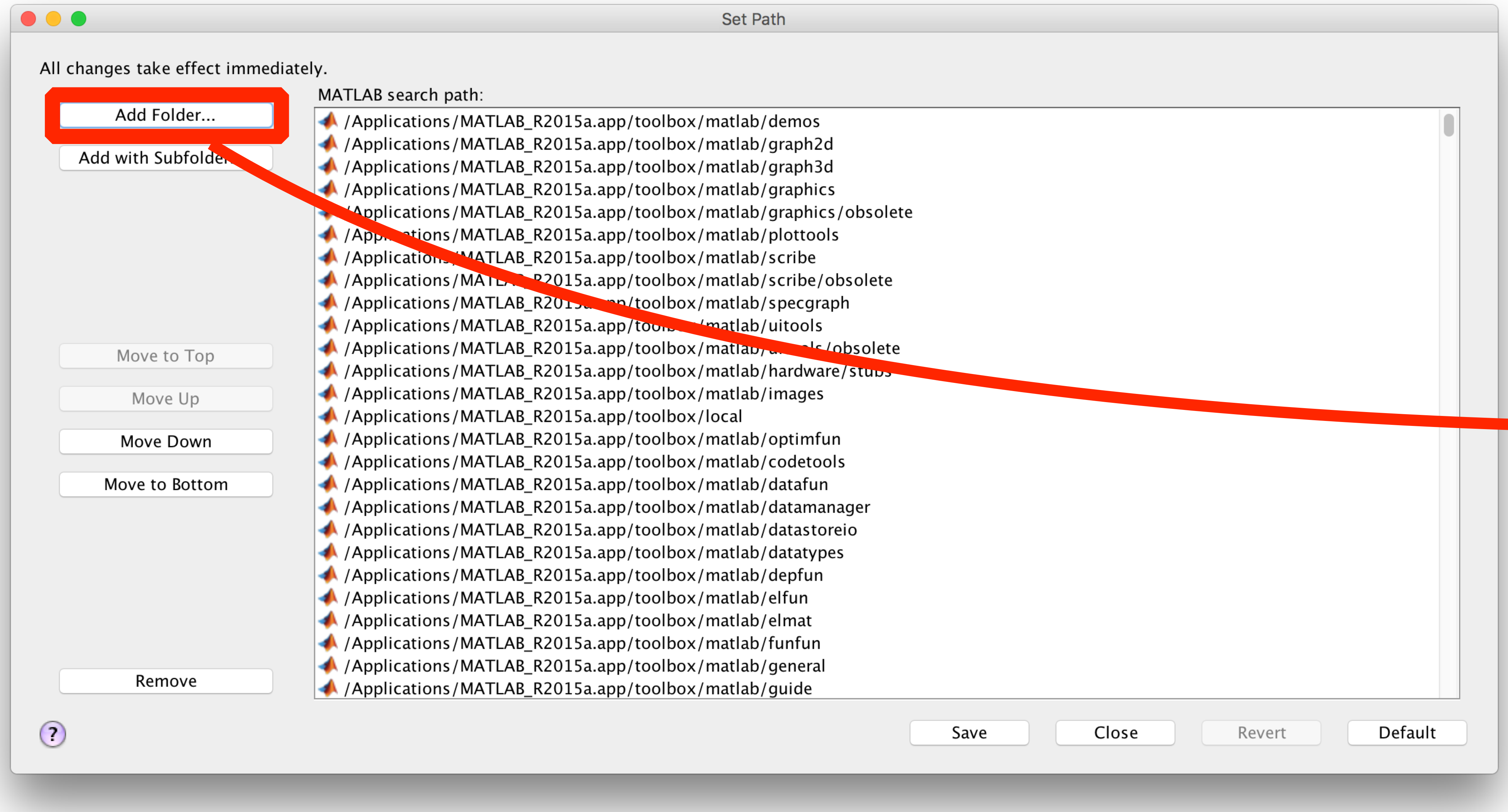
Matlab lecture 3 (Early January, 2017):
Bayesian theory and effective connectivity in SPM
at least 3 hours

Contents

- Set path
- Variables and basic operations
- Useful built-in functions
- User-define function
- Analysing E-prime output with Matlab

Set Path to use functions everywhere





All changes take effect immediately.

MATLAB search path:

```
/Applications/MATLAB_R2015a.app/toolbox/matlab/demos
/Applications/MATLAB_R2015a.app/toolbox/matlab/graph2d
/Applications/MATLAB_R2015a.app/toolbox/matlab/graph3d
/Applications/MATLAB_R2015a.app/toolbox/matlab/graphics
/Applications/MATLAB_R2015a.app/toolbox/matlab/graphics/obsolete
/Applications/MATLAB_R2015a.app/toolbox/matlab/plottools
/Applications/MATLAB_R2015a.app/toolbox/matlab/scribe
/Applications/MATLAB_R2015a.app/toolbox/matlab/scribe/obsolete
/Applications/MATLAB_R2015a.app/toolbox/matlab/specgraph
/Applications/MATLAB_R2015a.app/toolbox/matlab/uitools
/Applications/MATLAB_R2015a.app/toolbox/matlab/uitools/obsolete
/Applications/MATLAB_R2015a.app/toolbox/matlab/hardware/stubs
/Applications/MATLAB_R2015a.app/toolbox/matlab/images
/Applications/MATLAB_R2015a.app/toolbox/local
/Applications/MATLAB_R2015a.app/toolbox/matlab/optimfun
/Applications/MATLAB_R2015a.app/toolbox/matlab/codetools
/Applications/MATLAB_R2015a.app/toolbox/matlab/datafun
/Applications/MATLAB_R2015a.app/toolbox/matlab/datamanager
/Applications/MATLAB_R2015a.app/toolbox/matlab/datastoreio
/Applications/MATLAB_R2015a.app/toolbox/matlab/datatypes
/Applications/MATLAB_R2015a.app/toolbox/matlab/depfun
/Applications/MATLAB_R2015a.app/toolbox/matlab/elfun
/Applications/MATLAB_R2015a.app/toolbox/matlab/elmat
/Applications/MATLAB_R2015a.app/toolbox/matlab/funfun
/Applications/MATLAB_R2015a.app/toolbox/matlab/general
/Applications/MATLAB_R2015a.app/toolbox/matlab/guide
```

Users can add folders, such as SPM12, eeglab, and some other toolbox, to the 'Path'.
After adding folders to 'Path', users can call functions everywhere.

or in command line,

```
>> addpath('/Users/skyeong/spm12')     % add spm12 folder
>> addpath('/Users/skyeong/eeglab')    % add eeglab folder

>> addpath('C:\spm12')     % add spm12 folder
>> addpath('C:\eeglab')    % add eeglab folder
```

Add path in the commend line works well.
However, you have to type this commend whenever you execute Matlab program.

Matlab Getting Started

help / to get howto information

```
>> help isfield
   isfield True if field is in structure array.
      isfield(S,FIELD) returns true if the string FIELD is the name of a
      field in the structure array S.

      TF = isfield(S,FIELDNAMES) returns a logical array, TF, the same size
      as the size of the cell array FIELDNAMES.  TF contains true for the
      elements of FIELDNAMES that are the names of fields in the structure
      array S and false otherwise.

      NOTE: TF is false when FIELD or FIELDNAMES are empty.

      Example:
         s = struct('one',1,'two',2);
         fields = isfield(s,{'two','pi','One',3.14})

      See also getfield, setfield, fieldnames, orderfields, rmfield,
      isstruct, struct.

      Other functions named isfield

      Reference page in Help browser
         doc isfield
```

what, pwd, cd, which

```
>> what        % show matlab codes in an alphabetic order
   MATLAB Code files in the current folder /Volumes/JetDrive/workshops/
   Matlab/lecture1/codes

   anal_Eprime          ex1                  find_column_number
   compute_eprime_data  ex2
>>
>> pwd                             % show present working directory
   ans =

   /Volumes/JetDrive/workshops/Matlab/lecture1/codes
>>
>> cd('/Users/skyeong/Desktop')   % change directory / mac or linux
>> cd('C:\Documents')              % change directory / windows
>>
>> which pwd               % locate ‘pwd’ command
>> which ('spm')           % locate ‘spm’ command
```

Variables and basic operations

Scalars and Vectors

```
>> a = 10;                    % integer
>> b = 10.1;                  % real number
>>
>> A_vec = [a, b, 41];        % row vector for real number
>> B_vec = [1, 2, 10]';       % column vector for real number
>>
>> c = a+b;                   % addition of a and b
>> sum(A_vec)                 % sum of elements in A_vec
>> C_sum = A_vec + B_vec ???  % Matrix dimension must agree
>> C_sum = A_vec + B_vec';    % addition of A_vec and B_vec
>> C_m = A_vec*B_vec;         % matrix multiplication (C_m is a scalar)
>> C_m2 = A_vec.*B_vec'       % multiplication in element by element (C_m2 is a vector)
>> C_d = A_vec./B_vec’        % divide in element by element (C_d is a vector)
```

Character and Cell

```
>> A = 'I am sunghyon';                    % char
>> B = ['I' 'am' 'sunghyon'];              % also, char
>> C = ['I' ' am' ' sunghyon'];            % also, char
>>
>> a = 'I';                                % char
>> b = 'am';                               % char
>> c = 'sunghyon';                         % char
>> name = [a, b, c];                       % char
>> name2 = [a, ' ' b, ' ' c];              % char
>>
>> subjlist = {'jjkim', 'shkyeong','ybshin'};     % cell
>> subjlist{1}                                    % jaejkim is returned / char
>> subjlist{2}                                    % shkyeong is returned / char
```

Struct (put data)

```
>> subj = struct();                  % initialising struct variable

>> subj(1).name = 'jjkim';           % put name field to the struct
>> subj(1).age = 40;                 % put age field to the struct
>> subj(1).sex = 'Male';             % put sex field to the struct
>> subj(1).position = 'Boss';        % put position field to the struct
>>
>> subj(2).name = 'shkyeong';        % put name field to the struct
>> subj(2).age = 33;                 % put age field to the struct
>> subj(2).sex = 'Male';             % put sex field to the struct
>> subj(2).position = 'Podoc';       % put position field to the struct
>> subj
```

Struct (get data)

```
>> isfield(subj(1),'name')         % check whether or not subj has name
>> isfield(subj(1),'salary')       % check whether or not subj has salary
>>
>> getfield(subj(1),'name')        % get name of the first subject
>> subj(1).name                    % simple way to get name of subj(1)
>> subj(1)('name')                 % another way to get name of subj(1)
>>
>> getfield(subj(2),'position')    % get name of the first subject
>> subj(2).position                % simple way to get position
>>
>> names = {subj.name};            % get names as cell variable
>> names = [subj.name];            % ?????? I don’t recommend this way
>> positions = {subj.position};    % get positions as cell variable
```

Matrix and its basic operation

```
>> A_mat = [1,2,3; 3,4,5; 7,8,9];  % 3x3 matrix
>> B_mat = [1,1,1; 2,2,2; 3,3,3];  % 3x3 matrix
>> B_vec = [1;2;3];                % 3x1 matrix
>> a_mat = [1,1; 2,2; 3,3];        % 2x2 matrix
>>
>> C1 = A_mat + B_mat;             % addition in element by element
>> C2 = A_mat * B_mat;             % matrix multiplication
>> C3 = A_mat .* B_mat;            % element by element multiplication
>>
>> D1 = A_mat * B_vec;             % matrix operation
>> D2 = A_mat .* B_vec;            % error
```

Useful built-in functions

fprintf / sprintf

- **fprintf** formats data and displays the results in the targeted file or on the screen.
- **sprintf** formats data and returns the results in a string.

```
>> subjlist = {'jjkim','skyeong','honghong'};       % subjlist / a cell variable
>>
>> fprintf('%d-th subj is %s\n', 1, subjlist{1});   % print results on display
>> fn_nii = sprintf('con_0001_%s.nii',subjlist{1}); % print format string
>>
>> % Writing a formatted output in a file
>> fid = fopen('example.csv','w+');                 % create file open handler.
>> fprintf(fid,'id, subjname, age, position\n');
>> fprintf(fid,'%d, %s, %d, %s\n', 1, subj(1).name, subj(1).age, subj(1).position);
>> fprintf(fid,'%d, %s, %d, %s\n', 2, subj(2).name, subj(2).age, subj(2).position);
>> fclose(fid);                                     % close file open
```

Continued… / % format

- `%s`: a string
- `%d`: integer / `%03d`: three digits integer (ex. 1 —> 001)
- `%f`: real number
- `%.1f (%.3f)`: real number with 1 (3) decimal(s) expression.

```
>> A = [3, 3, 4];                               % A vector / size of 1x3
>> fprintf('%d %01d %02d\n', A(1), A(2), A(3));
>> fprintf('%.1f %.2f %.3f\n', A(1), A(2), A(3));
>> fprintf('%d, %d, %d\n', A(1), A(2), A(3));
>> fprintf('%d-%d-%.1f\n', A(1), A(2), A(3));
>> fprintf('%d-%d-%.1f\n', A);                  % simply since A is a vector,
```

fileparts / fullfile

- **fileparts**(FILE) returns the path, file name, and file name extension for the specified FILE
- **fullfile**(DIR1,DIR2,…, filename) Build full file name from parts.

```
>> fn = '/Users/skyeong/Desktop/example.csv';           % specify file name
>> [p, f, e] = fileparts(fn);                           % path, file name, extension.
>>
>> data_path = '/Users/skyeong/Desktop';                % a char variable
>> fn1 = fullfile(data_path,'con_0001_jaejkim.nii');   % jjkim’s con_0001 file path
>> fn2 = fullfile(data_path,'con_0001_skyeong.nii');   % kyeong’s con_0001 file path
```

for, while / loop

- **for** and **while** are used when we want to do the same thing with changing subject. For example, Think about that computing functional connectivity for 100 subjects.

for-loop

```
>> % count No. of subj
>> nsubj = length(subj);
>>
>> % repeat commands within for-loop
>> for c=1:nsubj,
>>     fprintf('%d, %s.\n',c,subj(c).name);
>> end
```

while-loop

```
>> % count No. of subj
>> nsubj = length(subj);
>>
>> % repeat commands within for-loop
>> cnt = 1;
>> while 1,
>>     fprintf('%d, %s.\n',cnt,subj(cnt).name);
>>     if cnt==nsubj,
>>         break;
>>     end
>>     cnt = cnt+1;
>> end
```

matrix operation vs. loop

- Matlab has a great advantage when using matrix operation.
- Let’s assume that we have dose and days of drug administration as following:

```
>> dose = [30, 20.1, 40, 22, 50, 10, 22, 23, 30, 21, 40, 44];
>> days = [32, 21, 11, 30, 16, 2, 30, 4, 11, 2, 36, 59];
```

- What is the total amount of drug administration (dose x days) ?

using for-loop

```
>> total_amount = zeros(1,12);
>>
>> for i=1:12,
>>     total_amount(i) = dose(i)*days(i);
>> end
```

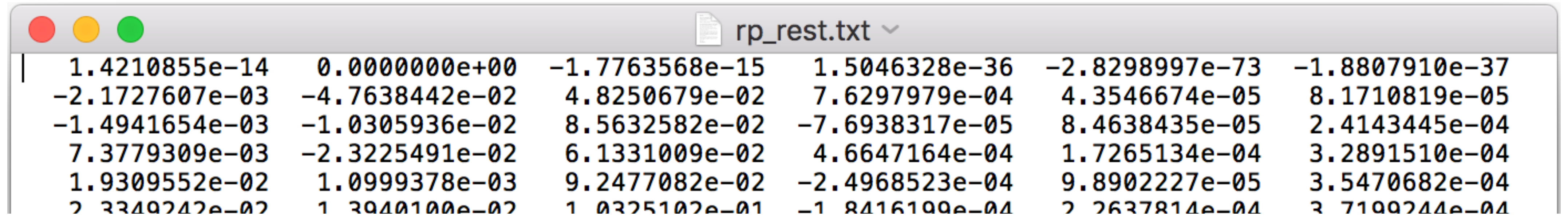
using matrix operation

```
>> total_amount2 = dose.*days;
```

Reducing **lines** and computation **times**

dlmread / dlmwrite

- **dlmread** and **dlmwrite** are used when we want to read and write numeric data (csv-like file but only for numeric data) with a specific separator such as a comma (,), semicolon (;), or tab (\t).



```
  1.4210855e-14   0.0000000e+00  -1.7763568e-15   1.5046328e-36  -2.8298997e-73  -1.8807910e-37
 -2.1727607e-03  -4.7638442e-02   4.8250679e-02   7.6297979e-04   4.3546674e-05   8.1710819e-05
 -1.4941654e-03  -1.0305936e-02   8.5632582e-02  -7.6938317e-05   8.4638435e-05   2.4143445e-04
  7.3779309e-03  -2.3225491e-02   6.1331009e-02   4.6647164e-04   1.7265134e-04   3.2891510e-04
  1.9309552e-02   1.0999378e-03   9.2477082e-02  -2.4968523e-04   9.8902227e-05   3.5470682e-04
  2.3349242e-02   1.3940100e-02   1.0325102e-01  -1.8416199e-04   2.2637814e-04   3.7199244e-04
```

```
>> help dlmread
>> help dlmwrite
```

```
>> DATApath = '/Users/skyeong/Desktop/data';
>>
>> % Load Realignment Parameters
>> fn_motion = fullfile(DATApath,'rp_rest.txt');
>> MOTION = dlmread(fn_motion);
>>
>> % Check size and elements of MOTION
>> size(MOTION)
>> MOTION(:,1:3)      % show first three column
>> MOTION(:,4:end)    % show 4-th column to the last
```

```
>> % Save first three column to the separate file
>> fn_out1 = fullfile(DATApath,'rp_rest_trans.csv');
>> TRANS = MOTION(:,1:3);
>> dlmwrite(fn_out1, TRANS);
>>
>>
>> % Save first three column to the separate file
>> fn_out1 = fullfile(DATApath,'rp_rest_rot.txt');
>> ROT = MOTION(:,4:6);
>> dlmwrite(fn_out1, ROT, 'delimiter','\t');
```

figure / plot

hands-on: **ex1**.m

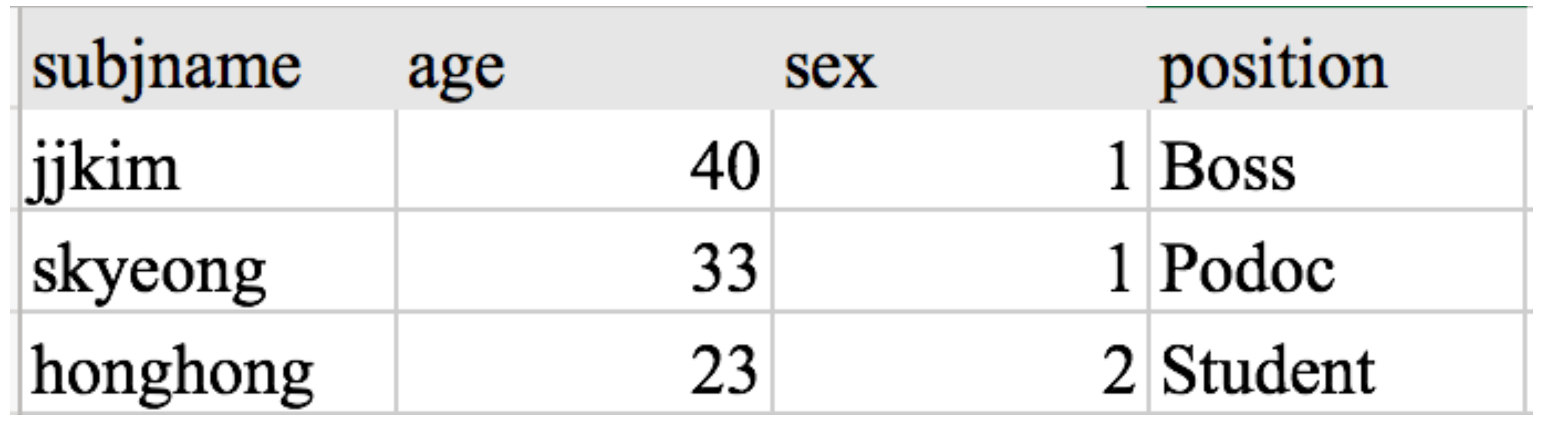
```
>>  DATApath = '/Users/skyeong/Desktop/data';
>>
>>  % Load Realignment Parameters
>>  fn_motion = fullfile(DATApath,'rp_rest.txt');
>>  MOTION = dlmread(fn_motion);
>>  size(MOTION);    % Check size of variable
>>  scans = 3:152;
>>
>>  % Split translation and rotation part
>>  TRANS = MOTION(:,1:3);       % in mm
>>  ROT = 50*MOTION(:,4:end);    % l=r*theta (r=5cm)

>>  % Plot Head Motion - translation
>>  figure;
>>  plot(scans,TRANS(scans,:));
>>  xlabel('Scan number');
>>  ylabel('Translation, mm');
>>  legend('x','y','z');
```

```
>>  % Plot Head Motion - rotation part
>>  figure;
>>  plot(scans,ROT(scans,:));
>>  xlabel('Scan number');
>>  ylabel('Rotation, mm');
>>  legend('pitch','roll','yaw');

>>
>>  % Plot Head Motion - translation part
>>  figure;
>>  subplot(211); plot(scans,TRANS(scans,:));
>>  xlabel('Scan number');
>>  ylabel('Translation, mm');
>>  legend('x','y','z');
>>
>>  % Plot Head Motion - rotation part
>>  subplot(212); plot(scans,ROT(scans,:));
>>  xlabel('Scan number');
>>  ylabel('Rotation, mm');
>>  legend('pitch','roll','yaw');
```

xlsread / to read data from excel file

hands-on: **ex2**.m

Excel data:



subjname	age	sex	position
jjkim	40	1	Boss
skyeong	33	1	Podoc
honghong	23	2	Student

```
>> DATApath = '/Users/skyeong/Desktop/data';
>>
>> % Load Excel data
>> fn_xls = fullfile(DATApath,'subjlist.xlsx');
>> [a,b,xlsData] = xlsread(fn_xls);
>>
>>
>> % Separate header and data
>> hdrs = xlsData(1,:);
>> data = xlsData(2:end,:);
>>
>>
>> % Get list of subjname
>> col_subjname = find_column_number(hdrs,'subjname');
>> list_subject = data(:,col_subjname);
```

```
>> % Get list of position
>> col_position = find_column_number(hdrs,'position');
>> list_position = data(:,col_position);
>>
>>
>> % Get list of age
>> col_age = find_column_number(hdrs,'age');
>> list_age = data(:,col_age);
>> list_age = cell2mat(list_age);
>>
>>
>> % Get list of sex
>> col_sex = find_column_number(hdrs,'sex');
>> list_sex = cell2mat(data(:,col_sex));
```

User-defined function (1)

hands-on: **find_column_number**.m

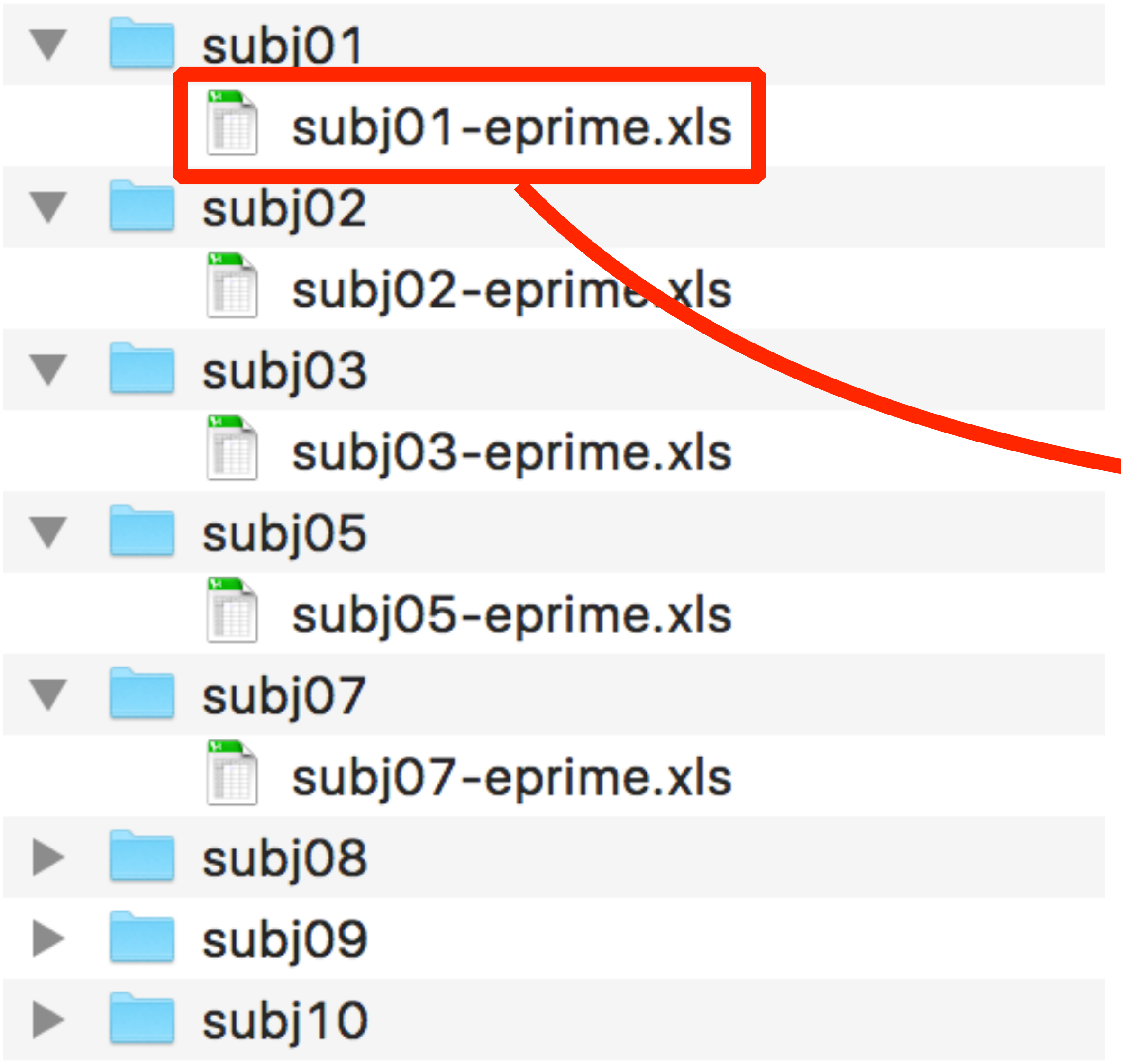
```
function column = find_column_number(list_of_headers,name_of_field)
%FIND_COLUMN_NUMBER is to find the column number from headers
%
%   FIND_COLUMN_NUMBER(list_of_headers, name_of_field) locates the column number
%   of name_of_field.
%
%   list_of_headers - {'name','age','sex','position'};
%   name_of_field   - 'name' or 'position'
%
%   Example:
%     list_of_headers = {'name','age','sex','position'};
%     find_column_number(list_of_headers, 'age');
%     return value would 2 because 'age' is located at the second.

for i=1:length(list_of_headers)
    if strcmpi(list_of_headers{i},name_of_field),
        colnum = i;
        return
    end
end
```

```
>>help strcmpi
>>help find_column_number
```

Analysing
E-prime output data with Matlab

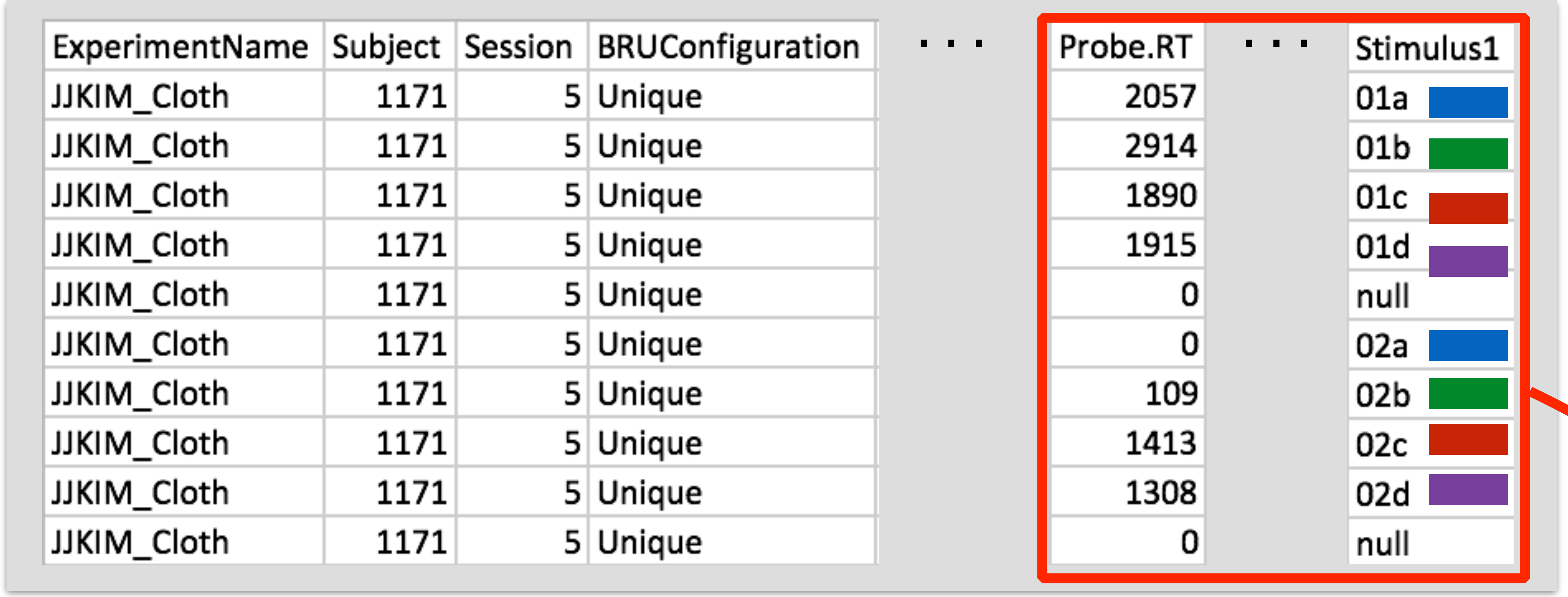
Work flow…

E-prime for each subject

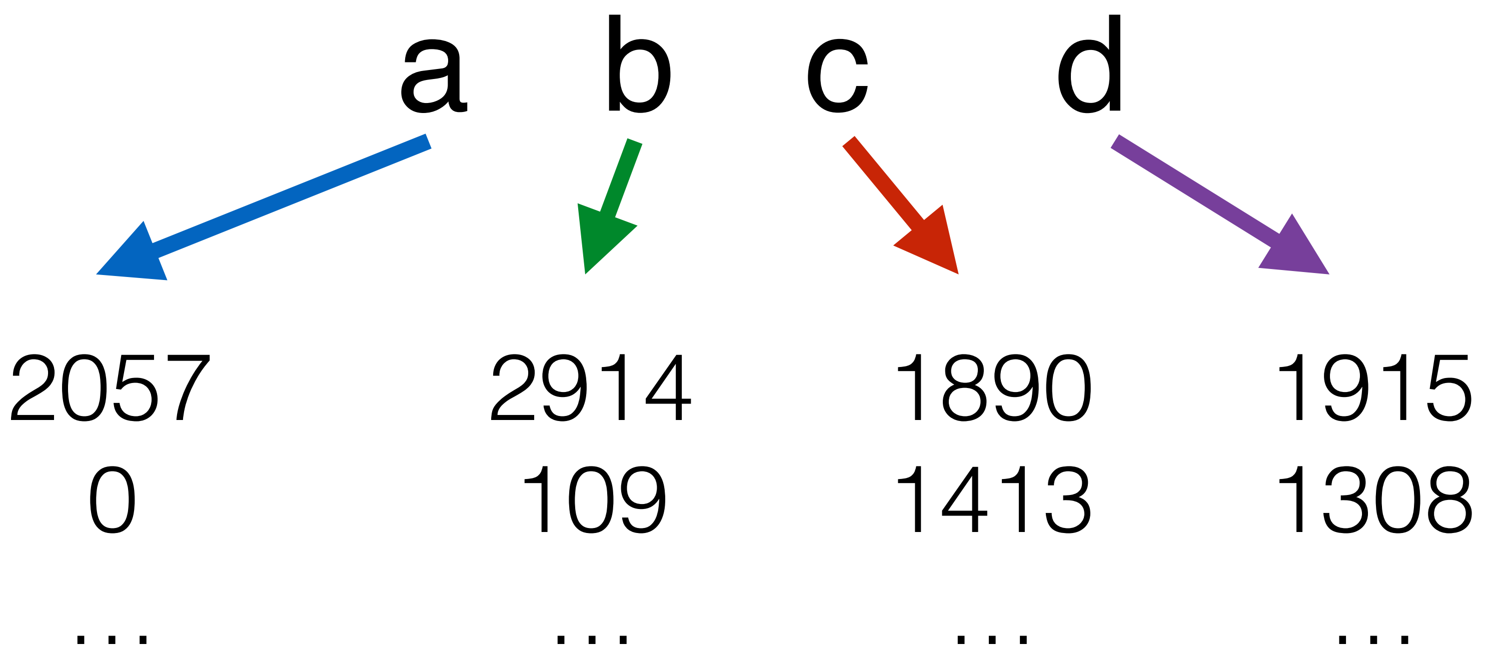


- subj01
  - subj01-eprime.xls
- subj02
  - subj02-eprime.xls
- subj03
  - subj03-eprime.xls
- subj05
  - subj05-eprime.xls
- subj07
  - subj07-eprime.xls
- subj08
- subj09
- subj10

E-prime output data looks like…



ExperimentName	Subject	Session	BRUConfiguration	· · ·	Probe.RT	· · ·	Stimulus1
JJKIM_Cloth	1171	5	Unique		2057		01a 🟦
JJKIM_Cloth	1171	5	Unique		2914		01b 🟩
JJKIM_Cloth	1171	5	Unique		1890		01c 🟥
JJKIM_Cloth	1171	5	Unique		1915		01d 🟪
JJKIM_Cloth	1171	5	Unique		0		null
JJKIM_Cloth	1171	5	Unique		0		02a 🟦
JJKIM_Cloth	1171	5	Unique		109		02b 🟩
JJKIM_Cloth	1171	5	Unique		1413		02c 🟥
JJKIM_Cloth	1171	5	Unique		1308		02d 🟪
JJKIM_Cloth	1171	5	Unique		0		null



collect RT data
for each stimulus type

anal_Eprime.m (1/3)

```
>> DATApath = '/Users/skyeong/Matlab/lecture1/data';
>>
>> % Load Subject List
>> fn_subjlist = fullfile(DATApath,'subjlist_eprime.xlsx');
>> [a,b,xlsData] = xlsread(fn_subjlist);
>> subjlist = xlsData(2:end,1);
>> nsubj = length(subjlist);  % count the cell element in 'subject' variable
>>
>> % Calculate average RT for each subject and condition
>> RT = struct();
>>
>> for c=1:nsubj,
>>
>>     subjname = subjlist{c};
>>     fprintf('analyzing data for %s.\n',subjname);
>>
>>     % Load Eprime data for each subject
>>     filename = sprintf('%s-eprime.xls',subjname);
>>     fn_xls = fullfile(DATApath,'Eprime',subjname,filename);
>>     [a,b,xlsData] = xlsread(fn_xls);
```

anal_Eprime.m (2/3)

```
>>
>>     % Spliting headers and data
>>     hdrs = xlsData(1,:);
>>     data = xlsData(2:end,:);
>>
>>     % Get colnum IDs for RunTitle, RESP, RT, and StimType
>>     colnum_RT       = find_column_number(hdrs,'Probe.RT');
>>     colnum_evtType  = find_column_number(hdrs,'Stimulus1');
>>
>>     % Get session information, RESP, RT, evtType
>>     Eprime = struct();
>>     Eprime.RT       = cell2mat(data(:,colnum_RT));
>>     Eprime.evtType  = data(:,colnum_evtType);
>>
>>
```

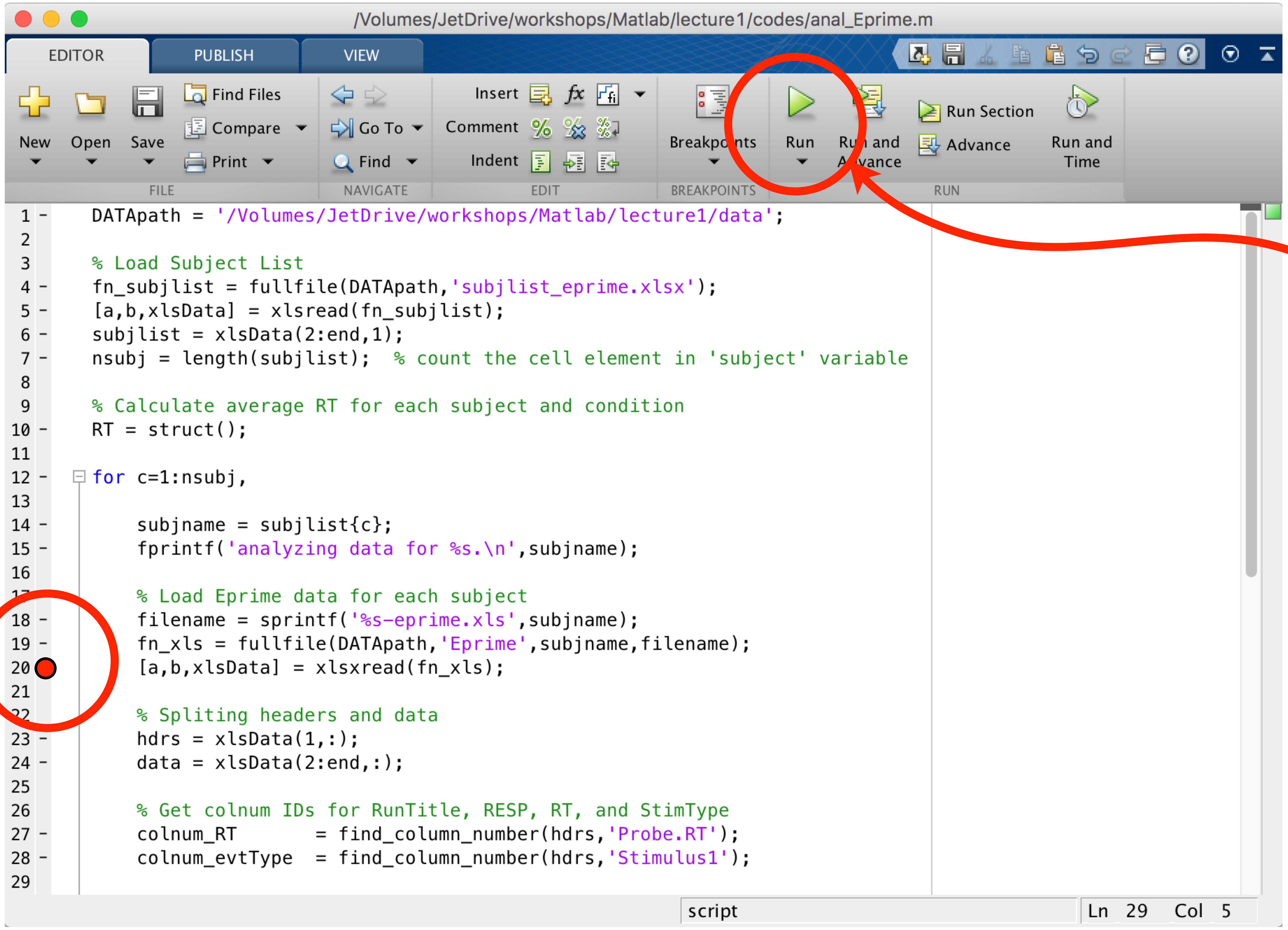
anal_Eprime.m (3/3)

```
>>  % Catagorizing Eprime data for each condition
>>     RT_subj = compute_eprime_data(Eprime);
>>     evtTypes = fields(RT_subj);
>>     for i=1:length(evtTypes),
>>         evtType = evtTypes{i};
>>         if strcmpi(evtType,'l'), continue; end
>>         RT(c).(evtType)   = mean([RT_subj.(evtType)]);
>>     end
>> end
>>
>>
>> % Write results in a csv-file
>> fn_out = fullfile(DATApath,'anal_eprime.csv');
>> fid    = fopen(fn_out,'w+');
>> fprintf(fid,'subjname,RT.a,RT.b,RT.c,RT.d\n');
>> for c=1:nsubj,
>>     fprintf(fid,'%s,%.1f,%.1f,%.1f,%.1f\n',subjlist{c},RT(c).a,RT(c).b,RT(c).c,RT(c).d);
>> end
>> fclose(fid);
```

Debugging a Matlab Program

Set breakpoints at each line

Click here
to set a breakpoint

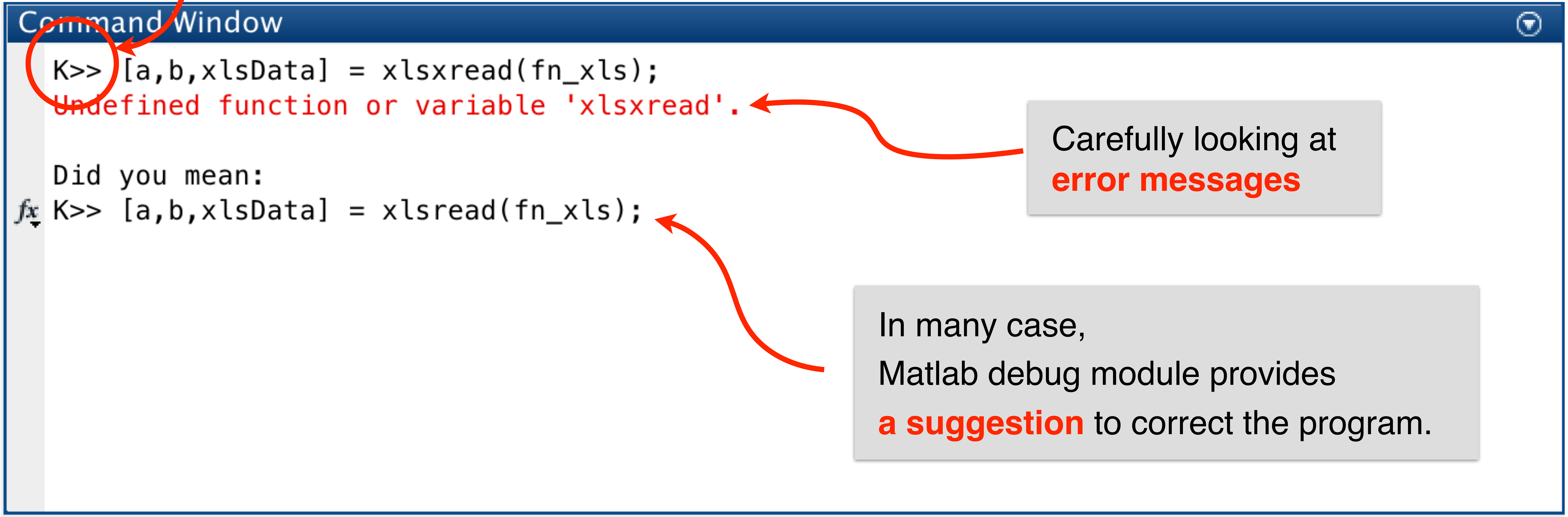


```
DATApath = '/Volumes/JetDrive/workshops/Matlab/lecture1/data';

% Load Subject List
fn_subjlist = fullfile(DATApath,'subjlist_eprime.xlsx');
[a,b,xlsData] = xlsread(fn_subjlist);
subjlist = xlsData(2:end,1);
nsubj = length(subjlist);  % count the cell element in 'subject' variable

% Calculate average RT for each subject and condition
RT = struct();

for c=1:nsubj,

    subjname = subjlist{c};
    fprintf('analyzing data for %s.\n',subjname);

    % Load Eprime data for each subject
    filename = sprintf('%s-eprime.xls',subjname);
    fn_xls = fullfile(DATApath,'Eprime',subjname,filename);
    [a,b,xlsData] = xlsread(fn_xls);

    % Spliting headers and data
    hdrs = xlsData(1,:);
    data = xlsData(2:end,:);

    % Get colnum IDs for RunTitle, RESP, RT, and StimType
    colnum_RT       = find_column_number(hdrs,'Probe.RT');
    colnum_evtType  = find_column_number(hdrs,'Stimulus1');

```

Run
a script

Debug mode in command window

“**K>>**” indicates that a program is in debug mode

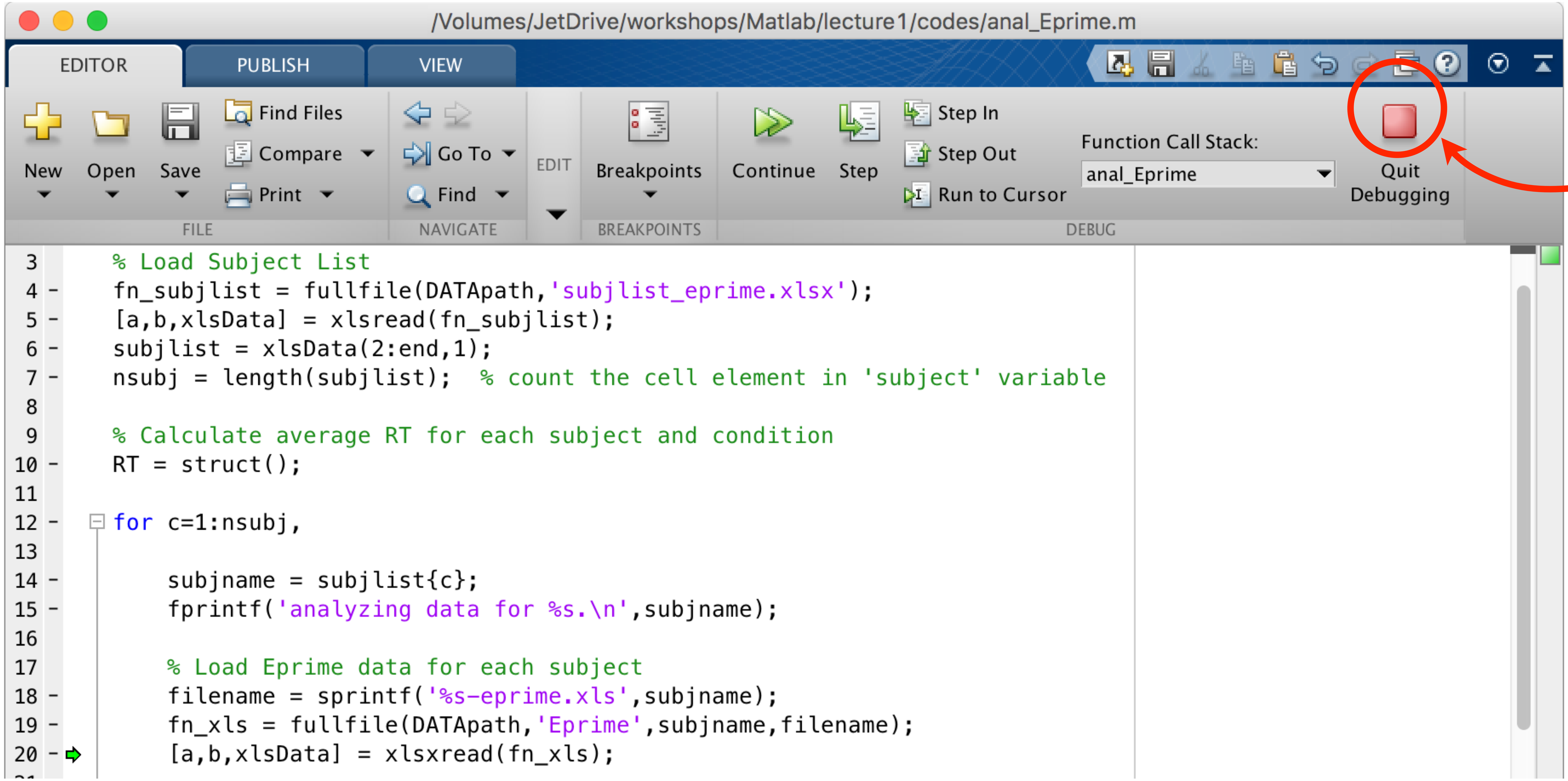


```
K>> [a,b,xlsData] = xlsxread(fn_xls);
Undefined function or variable 'xlsxread'.

Did you mean:
K>> [a,b,xlsData] = xlsread(fn_xls);
```

Carefully looking at
error messages

In many case,
Matlab debug module provides
a suggestion to correct the program.

To escape from a debug mode



```matlab
% Load Subject List
fn_subjlist = fullfile(DATApath,'subjlist_eprime.xlsx');
[a,b,xlsData] = xlsread(fn_subjlist);
subjlist = xlsData(2:end,1);
nsubj = length(subjlist);  % count the cell element in 'subject' variable

% Calculate average RT for each subject and condition
RT = struct();

for c=1:nsubj,

    subjname = subjlist{c};
    fprintf('analyzing data for %s.\n',subjname);

    % Load Eprime data for each subject
    filename = sprintf('%s-eprime.xls',subjname);
    fn_xls = fullfile(DATApath,'Eprime',subjname,filename);
    [a,b,xlsData] = xlsread(fn_xls);
```

Quit debugging