```
Error while evaluating this code:

% Sparky
   clear all; %remove all variables
clc %clear the command window
close all


close all%close the figures
% Load the output file from Imaris.
% Tell the matlab code where the first line and column of the stepData are.
%
% Remember text cells before number cells (in xlsread spreadsheets) are ignored
% but text cells after number cells are not.
top=1;
left = 1;


% disp('Do you want to accept these defaults?')
% disp( 'time between timepoints - 30 seconds');
% disp('time between distance measurements - 180 seconds');
% disp('flag the channel 2 Ca signal if below -40');
% disp('*')
% disp('*')
%     question = input('Y or N?','s'); %this is a text variable
% if question =='N' | question=='n' % ask for the values.
% disp('*')
%   timepointInterval=input('Enter the time between timepoints (seconds). --');
%      distanceInterval= input ('Enter the time between distance measurements (seconds). ↙
--');
%      lowfluorFlag = input ('Flag the channel 2 Ca signal if below -');
% else %otherwise, these are the defaults
%     disp('yes')
% timepointInterval= 30;
% distanceInterval= 180;
% lowfluorFlag=40;
% end
% disp('*')
% disp('*')
% disp('*')
%     CaCorrection = input('enter the Ca correction--');
%



%put in the correct path for the CaResultsTemplate
SparkyTemplatePath=pwd;
SparkyResultsTemplate= strcat(SparkyTemplatePath,'\Sparky results template.xls');


disp('LoadStepData')
disp('Get the spreadsheet')
% defaultPath='F:\Matlab\differing distances\data'
defaultPath= strcat(SparkyTemplatePath, '\data\');
saveName='LoadTrackDataSaveName';
  [imageFileNames, imagePathName] = getImageNamesFunction(defaultPath,saveName);
  RootPosition = findstr(imageFileNames, '.xls');    % the root name ends just before  .↙
xls
    %RootPosition = findstr(imageFileNames, '.txt');    % the root name ends just before↙
.txt
  RootName = imageFileNames (1:RootPosition-1);
SparkyResults= strcat(imagePathName,RootName, ' SparkyResults.xls');
%Read the Excel file
stepData= xlsread(strcat(imagePathName,imageFileNames),'Sheet1');
% stepData= dlmread(strcat(imagePathName,imageFileNames),'\t');
specs = xlsread(strcat(imagePathName,imageFileNames),'Sheet2');
if size(specs,1)>=7
    display('The data on sheet2 is incorrect')
 return
end
timepointInterval= specs(1,1);
distanceInterval= specs(1,2);
lowfluorFlag=specs(1,3);
CaCorrection = specs(1,4);
if stepData(1,1)>999999999
stepData(:,1)=stepData(:,1)-10^9;%remove the excess digits if they are there.
end

%initiate stepData
%stepData here is size= :,7
lengthStepData =size (stepData,1);
stepData(:,8:16)=zeros(lengthStepData,16-7); %columns 7-16 are zeros now

%initiate trackData
trackDataUnique =unique(stepData(:,1));%list of the tracks
trackData = zeros(size(trackDataUnique,1),19); %initialize
trackData (:,1)=trackDataUnique;

stepData(:,12) = stepData(:,6)./ stepData(:,7); %calculate Ca ratio
stepData(:,13) = stepData(:,12)-CaCorrection;
stepData(find(stepData(:,13)>.2),14) = 1;
%find the beginning and ending timepoints for each track
  for i=1:size(trackData,1)
   trackData(i,2)=min(find(stepData(:,1)==trackData(i,1)));
   trackData(i,3)=max(find(stepData(:,1)==trackData(i,1)));
  end

%find the differing distances
  for i=1:size(trackData,1)
 %  for j=1:6   %j determines the gap size
 delayTimepoints = distanceInterval/timepointInterval;
 writeStepColumn = 8; %initialize;
 writeTrackColumn = 4;%initialize;
 timeInterval = timepointInterval;%initialize;

 for j = [1 delayTimepoints]

    startingLine = trackData(i,2);
    endingLine  =  trackData(i,3);

    % this gets the distances with the different gaps (j)
  k=startingLine:endingLine-j ;
stepData (k,writeStepColumn)=((stepData (k,3)-stepData(k+j,3)).^2  +   ...    %X
               (stepData (k,4)-stepData(k+j,4)).^2  +   ...    %Y
               (stepData (k,5)-stepData(k+j,5)).^2  ).^.5   ;    %Z
  stepData (k,writeStepColumn+1) = stepData (k,writeStepColumn) .*60/ timeInterval; %↙
calculate the speed

  %make the extra steplines not a number (nan)
  stepData (endingLine+1-j:endingLine,[writeStepColumn writeStepColumn + 1])=nan;

%find track averages
trackData(i,writeTrackColumn)  = mean ( stepData(startingLine:endingLine-j,↙
writeStepColumn));%average distance
trackData(i,writeTrackColumn+1)  = mean ( stepData(startingLine:endingLine-j,↙
writeStepColumn+1));%average speed

writeStepColumn = writeStepColumn+2; %second time through write it to column 6
writeTrackColumn = writeTrackColumn+2; %second time through write it to column 8
 timeInterval = distanceInterval;
  end %(j)

  %get maximum displacement
m = startingLine+1:endingLine;
sizeM = size(m,2);
displacements(1:sizeM) =((stepData (startingLine,3)-stepData(m,3)).^2  +   ...    %X
               (stepData (startingLine,4)-stepData(m,4)).^2  +   ...    %Y
               (stepData (startingLine,5)-stepData(m,5)).^2  ).^.5   ;    %Z

trackData(i,8) = max(displacements);
  trackData (i,9) = mean (stepData(startingLine:endingLine,12)); %average Ca ratio
trackData(i,10)  = mean (stepData(startingLine:endingLine,13)); % average corrected Ca↙
ratio
trackData(i,11)  = max (stepData(startingLine:endingLine,13)); % max corrected Ca ratio
trackData(i,12)  = max (stepData(startingLine:endingLine,10));% maximum distance of the↙
delayed time interval
  end %(i)
trackData(:,13)  = trackData(:,3) -  trackData(:,2) +1; %how many timepoints for this↙
track

%find the pathlength
 k=1:size(trackData,1);
for i=k;
trackData(i,14) = sum(stepData(trackData(i,2):trackData(i,3)-1,8));
end
```

```
%final displacement
for i=k;
 trackData (i,15)=((stepData(trackData (i,2),3)-stepData(trackData (i,3),3)).^2  +   ...↙
%X
                   (stepData(trackData (i,2),4)-stepData(trackData (i,3),4)).^2   +   ...↙
%Y
                   (stepData(trackData (i,2),5)-stepData(trackData (i,3),5)).^2 ) ^.5   ;↙
%Z
end
trackData(k,16)=trackData(k,15)./trackData(k,14); % get displacement index

for count=1:size(trackData,1)
if     trackData(count,11)<= 0.2       %max corrected Ca <= 0.2   "LOW"
trackData(count,17)=1;
elseif  trackData(count,11)> 0.2  &... %max corrected Ca >0.2 AND
       trackData (count,10)> 0.1       % average corrected Ca >0.1   "HIGH"
trackData(count,17)=2;
elseif  trackData(count,11)> 0.2  &... %max corrected Ca >0.2 AND
       trackData (count,10)<=0.1       % average corrected Ca <=0.1   "OTHER"
trackData(count,17)=3;
end
stepData(trackData(count,2):trackData(count,3)  ,15) = trackData(count,17);
trackData(count,18)=sum(stepData(trackData(count,2):trackData(count,3)  ,14)) ; %↙
timepoints above cutoff
trackData(count,19)=trackData(count,18)/(trackData(count,3)-trackData(count,2)+1); %↙
fractions of timepoints that are above the cutoff
end

trackDataSize=size (trackData);
firstTP = 2;
lastTP = 3;
disp('Now do slowBuildPeaks')
            slowBuildPeaks %script
disp('Now do signalingPeakData')
            signalingPeakData %script
 disp('Now do events')
             Events   %script


% disp('Now do analyzeGroups')
%             analyzeGroups    %script
% disp('Now do moreGroupStats')
%             moreGroupStats   %script




            disp ('**********')
            disp('writing')
               try
 %remove the old file
 delete (SparkyResults)
```

```
copyfile(SparkyResultsTemplate,SparkyResults)
               catch
    close
    disp(SparkyResults)
    input('Close the SparkyResults  excel worksheet!  Then press return.')
%remove the old file
delete (SparkyResults)
copyfile(SparkyResultsTemplate,SparkyResults)
end
disp ('writing stepData')
  xlswrite(SparkyResults,timepointInterval,'stepData','E1')
  xlswrite(SparkyResults,distanceInterval,'stepData','M4')
  xlswrite(SparkyResults,lowfluorFlag,'stepData','J2')
  xlswrite(SparkyResults,CaCorrection,'stepData','P2')
  xlswrite(SparkyResults,stepData,'stepData','d6')
  xlswrite(SparkyResults,specs(1,5),'stepData','q5')
disp ('writing trackData')
  xlswrite(SparkyResults,trackData,'trackData','e6')
disp ('writing peakData')
  xlswrite(SparkyResults,peakData,'peak data','d6')

if noPeakDataSignaling==1
        disp('*')
    disp('*')
    disp('There may be no peaks in peakDataSignaling as defined in the signalingPeakData↙
script!')
    disp('*')
    disp('*')
else
disp ('writing peakDataSignaling')
   xlswrite(SparkyResults,peakDataSignaling,'peakDataSignaling','d6')
end
if isempty(events)
        disp('*')
    disp('*')
    disp('There may be no events!')
    disp('*')
    disp('*')
else
   disp ('writing events')
    xlswrite(SparkyResults,events,'events','d6')
end

  disp('Your results file is:')
  disp(SparkyResults)
if   plotConditions == 0
    close
end

Error using ==> xlsread at 123
Filename must be specified.
```

```
Warning: Could not find an exact (case-sensitive) match for
'SaveAs'.
C:\Program Files\MATLAB\R2007b\toolbox\matlab\general\saveas.m is
a case-insensitive match and will be used instead.
You can improve the performance of your code by using exact
name matches and we therefore recommend that you update your
usage accordingly. Alternatively, you can disable this warning
using
warning('off','MATLAB:dispatcher:InexactMatch').
> In codetools\private\mxdom2word at 159
  In publish at 215
  In mdbpublish at 53
??? Error using ==> saveas at 72
Undefined function or method 'get_param' for input arguments of
type 'Interface.Microsoft_Word_14.0_Object_Library._Document'.

Error in ==> mxdom2word at 159
    doc.SaveAs(outputPath,wdFormatDocument);

Error in ==> publish at 215
        mxdom2word(dom,outputAbsoluteFilename);

Error in ==> mdbpublish at 53
outputPath = publish(file, options);

>>
```