**Data Analysis Tools Documentation**

The github repository at <https://github.com/JaneliaSciComp/G4_Display_Tools> has a suite of data analysis tools.

In the typical use case of these tools, there are two files you need to worry about: “DA\_plot\_settings.m” at G4\_Display\_Tools/G4\_Data\_Analysis and “get\_exp\_folder.m” at G4\_Display\_Tools/G4\_Data\_Analysis/Support.

Table of Contents:

1. [New Format](#newFormat)
2. [Plot appearance settings](#appearanceSettings)
3. [Data analysis settings](#dataSettings)
4. [Running a typical analysis](#running)
5. [Adding new modules](#addModule)

**New Format:**

The files G4\_Process\_data\_flyingdetector (in DA\_Data\_Analysis/data processing) and G4\_Plot\_Data\_Combined (in DA\_Data\_Analysis/data plotting) have been changed slightly from previous versions. The processed .mat file produced is formatted differently, and this format is required to work with the data analysis tools described below.

For your convenience, there is a script called “convert\_processed\_file.m” in DA\_Data\_Analysis/support which will convert older processed files to this new format. Be sure to open this file and update the processed\_file\_name variable. If this does not match the filename of the processed file you want to convert (excluding the extension), it will not work.

You can run the convert\_processed\_file.m file and it will prompt you to browse to the experiment’s results folder. Alternatively you can pass this in by typing “convert\_processed\_file(folderpath)” in the matlab command line. You can simply pass in the path to your Results folder. The script will go through every fly folder and convert each processed file.

Alternatively, you may run the new processing file on old data to produce a new processed\_data file.

**Plot Appearance Settings:**

The first step to running analysis on your data is to ensure all settings are as you want them. Open “DA\_plot\_settings.m”.

The settings are split into eight different structs.

1. Exp\_settings.
   1. These are experiment settings and should be updated each time.
2. normalize\_settings.
   1. These are all settings related to the normalization of the data.
3. histogram\_plot\_settings.
   1. These are settings related to plotting basic histograms. These are not the same as the closed-loop histograms.
4. histogram\_annotation\_settings.
   1. These control how the histograms from 2 are annotated – font, line type, and many other things.
5. CL\_hist\_plot\_settings.
   1. These are the settings for closed-loop histograms
6. timeseries\_plot\_settings.
   1. Contains appearance settings for the timeseries plots
7. TC\_plot\_settings.
   1. Appearance settings for tuning curves
8. save\_settings.
   1. These settings affect how the results are saved.

The first section includes settings that you will likely need to update for every data analysis you do. The second section contains settings that will likely change less frequently, and mostly have to do with the details of plot appearance. Let’s go over the settings and what they mean.

1. Save\_path:
   1. A string indicating the path where you’d like your results saved.
2. Results\_path:
   1. A string indicating the path to the protocol folder which contains all your experiment results.
3. Genotypes:
   1. Usually results are being grouped by genotype and this would be an array of the names you want to give each genotype. However, you could group flies by categories other than genotype in which case this can be an array of labels corresponding to that category.
   2. EXAMPLE: if you have a results folder with the results from three different genotypes and you want to compare each genotype as a group to the others, you would set exp\_settings.genotypes = [“EmptySplit”, “LC-2”, “LPLC-15”] or whatever your genotypes are. Note that these do not have to match the exact genotype, they will only be used for labels and therefore should be human readable. If you were, however, grouping flies in some other way – like you want to compare all flies run by person A to all flies run by person B, this would be set equal to [“Person A”, “Person B”]
   3. NOTE: These strings should be enclosed in DOUBLE (“”) quotes.
4. Control\_genotype:
   1. If you would like to compare your genotypes (or groups) against a control, indicate in this variable which genotype is the control. The control genotype should be included in the “genotypes” variable, and should match the control\_genotype string exactly. The only difference being this string should be enclosed in SINGLE (‘’) quotes.
5. Processed\_data\_file:
   1. This is a variable of type string. It should be the name of your processed data files. For example, if the experiments you are analyzing have processed files called G4\_Data\_Processed.mat, then this variable should be set equal to ‘G4\_Data\_Processed’. Please make sure all fly folders in this analysis have their processed files named the same way. Any that are named differently will not be included in the analysis.
6. Group\_being\_analyzed\_name:
   1. A string indicating the name of the group being analyzed. This does not need to match any other files or variables so make it something recognizable to you. It will be used to name the group log file.
7. Date\_range:
   1. The date range over which these flies were collected to be annotated on histograms.
8. OL\_TS\_conds:
   1. This variable allows you to layout your timeseries plot figures exactly how you want them. If you are happy with the default layout, which will put 30 subplots on a figure in 6 rows x by 5 columns, then leave this variable empty (OL\_TS\_conds = [];)
   2. To layout the figures yourself, you will make OL\_TS\_conds a cell array, with each cell element representing one figure. Each cell array will contain a regular array of condition numbers laid out in the format of the subplots.
   3. EXAMPLE:   
      timeseries\_plot\_settings.OL\_TS\_conds{1} = [1 3 5 7; 9 11 13 15];  
      timeseries\_plot\_settings.OL\_TS\_conds{2} = [17 19; 21 23; 25 27; 29 31]  
        
      The above code will create two figures of timeseries plots, containing two rows of four subplots. The first figure will be laid out like:  
        
      Trial 1 Trial 3 Trial 5 Trial 7  
        
      Trial 9. Trial 11 Trial 13 Trial 15  
        
      The second figure, on the other hand, will contain four rows of two figures each. The first row will contain plots of condition 17 and 19, etc.   
        
      In this way, you can plot any condition in any position on the figure. Note that if you’re plotting more than one datatype, a figure is created for each datatype, so the above code, when run on two datatypes, would create four figures. Two for one datatype and two for the other.
9. OL\_TS\_durations:
   1. The x-axis limits of timeseries plots generally will be the duration of that condition. The OL\_TS\_durations array should match the OL\_TS\_conds array exactly in shape and size, but instead of containing the condition numbers, it should contain the corresponding durations of each condition.
   2. EXAMPLE: The durations array corresponding to the above OL\_TS\_conds array might look something like this:  
        
      timeseries\_plot\_settings.OL\_TS\_durations{1} = [1.5 3.12 1.5 3.12; 1.5 3.12 1.5 3.12]  
      timeseries\_plot\_settings.OL\_TS\_durations{2} = [1 3; 1 3; 1 3; 1 3];  
        
      …indicating that conditions 1, 5, 9 and 13 have durations of 1.5 seconds, 3, 7, 11, and 15 have durations of 3.12 seconds, 17, 21, 25, 29 have durations of 1 second, etc.
   3. If you left OL\_TS\_conds empty, leave this array empty as well. The program will deduce the condition durations from the processed data and create a default durations array.
10. OL\_TSconds\_axis\_labels:
    1. This variable contains the axis labels for timeseries plots in an array [x-label, y-label]. Each timeseries figure should get a set of labels, so to go with the above example it would look something like:   
       OL\_TSconds\_axis\_labels{1} = [“Time(sec)”, “LmR”];  
       OL\_TSconds\_axis\_labels{2} = [“Time(sec)”, “LmR”];
11. OL\_datatypes:
    1. This should be a cell array of the datatypes you which to plot timeseries data for. You may plot timeseries for as many or few datatypes as you like.   
         
       The datatype options for flying data are: ‘LmR\_chan’, ‘L\_chan’, ‘R\_chan’, ‘F\_chan’, ‘Frame Position’, ‘LmR’, ‘LpR’, and ‘faLmR’.  
         
       The datatype options for walking data are: ‘Vx0\_chan’, ‘Vx1\_chan’, ‘Vy0\_chan’, ‘Vy1\_chan’, ‘Frame Position’, ‘Turning’, ‘Forward’, and ‘Sideslip’
12. Show\_individual\_flies:
    1. This should be set to 0 or 1, 1 indicating that each individual fly should be plotted on the timeseries plots as well as the average. Default is 0. This should be set to 0 if plotting more than one group.
13. Frame\_superimpose:
    1. This should be set to 0 or 1, 1 indicating that the frame position should be plotted in light grey on the timeseries plots
14. Plot\_both\_directions:
    1. This should be set to 0 or 1, a 1 indicating that each timeseries subplot should plot the condition assigned to it as well as its corresponding condition of the opposite direction. An experiment utilizing two directions, like clockwise and counter clockwise, or up and down, should have all odd conditions be one direction and even directions another. If this variable is set to 1, your OL\_TS\_conds array should have all the odd conditions in it. The software will immediately add its even counter part to each axis. So for the OL\_TS\_conds array above, if this variable is set to 1, the first figure would look like:   
         
       Trial 1 and 2 Trial 3 and 4. Trial 5 and 6 Trial 7 and 8  
         
       Trial 9 and 10. Trial 11 and 12 Trial 13 and 14 Trial 15 and 16
15. Cond\_name:
    1. If you would like your timeseries subplots to all have their own titles, this should be an array, matching OL\_TS\_conds in shape and size, with each element being the plot title for that condition.
16. CL\_hist\_conds:
    1. This array is exactly the same as OL\_TS\_conds, except it will determine the layout of your closed loop histograms if you’re creating any. Leave empty if you are not plotting histograms or would prefer the default layout.
17. CL\_datatypes:
    1. Like OL\_datatypes, this is a cell array of all datatypes you wish to create closed loop histograms for.
18. OL\_TC\_conds:
    1. While this array is the same in purpose as OL\_TS\_conds and CL\_hist\_conds, it works a bit differently. This is because tuning curves plot multiple conditions on one axis.
    2. This is a two dimensional cell array. The first cell dimension is the number of figures. The second is the number of rows in that figure. Each cell element then contains a two dimensional array indicating which conditions should be included on the tuning curve and the tuning curve’s placement. It follows the format:   
       OL\_TC\_conds{fig #}{row #} = [condition #’s on first tuning curve; condition #’s on second tuning curve; …];
    3. EXAMPLE:  
       OL\_TC\_conds{1}{1} = [1 3 5 7; 9 11 13 15; 17 19 21 23]  
       OL\_TC\_conds{1}{2} = [ 25 27 29 31; 33 35 37 39; 41 43 45 47];  
       OL\_TC\_conds{2}{1} = [2 4 6 8; 10 12 14 16; 18 20 22 24];  
       OL\_TC\_conds{2}{2} = [26 28 30 32; 34 36 38 40; 42 44 46 48];  
         
       The above code creates two figures, each with two rows. The first figure is laid out as below:   
         
       TC w/ conds 1,3,5,7 TC w/ conds 9,11,13,15 TC w/ conds 17,19,21,23  
       TC w/ conds 25,27,29,31 TC w/ conds 33,35,37,39 TC w/conds 41,43,45,47
19. Xaxis\_label:
    1. TC\_plot\_settings.xaxis\_label is a string indicating the label of the x axis of the tuning curves. This should indicate what is changing between each condition on the curve.
20. Xaxis\_values:
    1. TC\_plot\_settings.xaxis\_values is an array of numbers indicating what values should be associated with each condition in the tuning curve on the xaxis. For example, if the conditions on your tuning curve are changing in frequency, then the x label would be frequency and the x values might be [10, 100, 500, 1000] indicating that the frequency of the first condition was 10 hz, the second was 100 hz, etc.
21. TC\_datatypes
    1. Like the other datatypes variables, this is a cell array of all the datatypes you wish to display as tuning curves.
22. Plot\_both\_directions
    1. Part of the TC\_plotting\_settings struct, this variable behaves exactly the same way as the timeseries version, but refers to the tuning curve plots.

This covers the settings that should be regularly updated. Below this section in “DA\_plot\_settings.m” you will find many more settings which mostly affect the appearance of the plots. You will find a full list with explanations in the appendix of this document.

Now you are ready to create your data analysis object and run an analysis!

**Running a typical analysis:**

There are two steps to running data analysis – the first is to run the file “create\_data\_analysis\_tool.m.” This is not a regular script or function, so opening the file and hitting run in the MATLAB environment will not work. It is a class and when you run it, it creates an object. You should run it from the matlab command line. Here’s an example:   
  
da = create\_data\_analysis\_tool(exp\_folder, trial\_options, ‘-group’, ‘-tsplot’);

The first required input is exp\_folder. This is a cell array of all paths to fly folders being analyzed. If there are multiple groups of flies (such as multiple genotypes) it will be a two dimensional cell array. For your convenience, there is a function which will create this for you called “get\_exp\_folder.m” located in G4\_Display\_Tools/G4\_Data\_Analysis/support. Let’s go over how to use this function.

When you open “get\_exp\_folder.m,” you’ll find six or so variables that need to be updated. Let’s go over what they are:

1. Field\_to\_sort\_by;
   1. This is a cell array, each element being a regular array. This allows you to only pull flies which match certain metadata values into your data analysis. If you want to sort your flies by genotype and only genotype, you would set this to:  
      field\_to\_sort\_by{1} = [“fly\_genotype”]; Notice that the string “fly\_genotype” must match exactly the genotype field name in your metadata.mat file.   
        
      field\_to\_sort\_by{1} = [“fly\_genotype”, “fly\_age”] means you will have one group of flies, narrowed down both by genotype and by age.   
        
      field\_to\_sort\_by{1} = [“fly\_genotype”];  
      field\_to\_sort\_by{2} = [“fly\_genotype”, “experimenter”];  
        
      The above means you will have two groups of flies. One group will be all flies that match a particular genotype. The second field will be all flies that match a particular genotype AND were run by a particular experimenter. These two groups will be plotted on your graphs for comparison.
2. Field\_values:
   1. This is where you provide the values of the above field to match. It is an array just like field\_to\_sort\_by, but in place of the field name you will put the value you want to match. So the corresponding field\_values for the examples above would look something like:   
        
      field\_values{1} = [“emptySplit\_JFRC100\_JFRC49]; The values you give here must match exactly the values in the metadata.mat file.   
        
      field\_values{1} = [“emptySplit\_JFRC100\_JFRC49”, “3-6 days”];  
        
      field\_values{1} = [“emptySplit\_JFRC100\_JFRC49”];  
      field\_values{2} = [“emptySplit\_JFRC100\_JFRC49”, “taylorl”];
3. Single\_group:
   1. Set this to 1 if you only want to plot a single group, 0 if you are plotting a single fly or multiple groups.
4. Single\_fly:
   1. Set this to 1 if you only want to analyze a single fly, 0 if multiple flies.
5. Trial\_options:
   1. This is a 3x1 array of zeros and 1’s indicating the presence of pre, inter, and post trials. [1 1 1] means all three were used. [1 0 1] means there was a pre and post trial but no intertrials.
6. Path\_to\_protocol:
   1. This is a string indicating the path to the protocol folder which contains all the flies to be considered for this analysis.

Once you have updated these variables, run [exp\_folder, trial\_options] = get\_exp\_folder() in your matlab command line. This will produce an exp\_folder cell array which has dimensions num\_groups x num\_experiments. It will also spit out the trial\_options variable. These two variables are the first two inputs in your data analysis command.

After exp\_folder and trial\_options there are multiple optional inputs, or flags, which tell the “create\_data\_analysis\_tool” function what analysis to do. The currently accepted flags are as follows:

* ‘-group’ – Include this if you’re analyzing many flies
* ‘-single’ – include this if you’re analyzing a single fly. NOTE: You must include either single or group flag!
* ‘-normfly’ – normalize the data over each fly
* ‘-normgroup’ – normalize the data over groups
* ‘-hist’ – plot basic histograms
* ‘-TSplot’ – plot open loop timeseries data
* ‘-CLhist’ – plot closed loop histograms
* ‘-TCplot’ – plot tuning curves

Make sure not to leave out the apostrophes or the dash. Any subset of these can be passed in, in any order. They are not case-sensitive.

When you run create\_data\_analysis\_tool, you want to store it in a variable. In the example above, I called this variable da.

This in itself will not run the analysis. What it does is creates an object, da, with all of your settings stored in it and the options for whatever flags you passed in turned on. You can use this object to double check if everything is correct if you would like. For example, you could now type “da.save\_settings” into the command line to review your save settings. If you forgot to pass in a flag, you could say da.TC\_plot\_option = 1 to retroactively tell it you want to make tuning curves as well. If you forgot to update the colors of your timeseries plot, you could say da.timeseries\_plot\_settings.rep\_colors = [0 0 0; 0 1 0; 0 0 1] and update them.

It is not likely you will want to update variables this way – it would be easier to open the DA\_plot\_settings file, edit it, and then run the command again. But when this tool needs to be called by other pieces of software, this system makes it much easier to automatically run the correct analysis without the software having to edit files (and possibly take you by surprise next time you use them!)

Once you know there are no adjustments to be made, simply type in the command da.run\_analysis, and this will start the analysis running. Assuming no adjustments after creating your data analysis tool, your commandline command will look something like this:

da = create\_data\_analysis\_tool(exp\_folder, trial\_options, ‘-group’, ‘-hist’, ‘-TSplot’, ‘-tcplot’)  
da.run\_analysis

This will produce a number of graphs, automatically saving them at the save path you entered. Their filenames will contain the genotypes being compared followed by a number.

**Adding new modules:**

Coming soon