**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. [Data](#newFormat) Processing
2. [Settings and what they mean](#appearanceSettings)
3. [Create settings file](#createSettings)
4. [Running a typical analysis](#running)
5. [Adding new modules](#addModule)
6. [Appendix](#appendix)

**Data processing:**

Before you can use the data analysis tools described below, you must process your data using either G4\_Process\_data\_flyingdetector or G4\_Process\_data\_flyingdetector\_combinedComm (both located in G4\_Data\_Analysis/data processing). The first file processes data which was collected using the separate Panel\_com commands, while the second processes data that was obtained using the new combined command. This step is done automatically after the experiment ends if you are using the G4 Conductor. It will produce a .mat file named G4\_Processed\_Data.

When running an experiment using the G4 conductor, there are two run protocols that you can use – the older, G4\_default\_run\_protocol.m and the newer G4\_run\_protocol\_combinedCommand.m. If you used the older run protocol, you must use the older processing file as well. Same with the new ones. If you mix and match these, your data processing will fail.

Note that there is an even older version of G4\_Process\_data\_flyingdetector. It saves all data in a single struct called Data rather than saving the data in multiple variables. If you have data that was processed using this old version, you will need to convert it to the new format before analyzing the data in it. 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.

Alternatively, you may run the current G4\_Process\_data\_flyingdetector file on old data to produce a new processed 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 structures.

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 will likely change with each type of analysis. The second section contains settings that will 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. Trial options:
   1. This is a 1x3 array of 0’s or 1’s indicating the presence or absence of a pre-trial, inter-trial, and post-trial. [1 1 1] – the experiment had all three. [0 1 0] – the experiment had inter trials, but no pre or post trial.
2. 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.
3. 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”]; all flies of both this genotype AND this age will be put in the first group.  
        
      field\_values{1} = [“emptySplit\_JFRC100\_JFRC49”];  
      field\_values{2} = [“emptySplit\_JFRC100\_JFRC49”, “taylorl”]; Will produce two groups of files to compare – first group has all flies of that genotype. Second group has all flies of that genotype AND run by that user.
   2. NOTE that if plot\_all\_genotypes is set to 1, field\_values can be left empty, because all values will be included. Likewise, if single\_fly is set to 1, field\_values should be empty.
4. 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
5. Single fly:
   1. Set this to 1 if you only want to analyze a single fly, 0 if multiple flies.
6. Plot all genotypes:
   1. This should be either 0 or 1. If it is set to 1, then each fly of a particular value of “field to sort by” will be placed in a group together. All values will be grouped. In addition, each group will be plotted individually against the control. If you have results for five genotypes and you set group1 as the control, you’ll end up with four sets of graphs – group1 v group2, group1 v group3, etc. Set this to 0 if you want to only include flies with a subset of values for your field to sort by, or if you want to put them on all one plot together rather than comparing to the control one by one.
7. Control genotype:
   1. This should be a character vector with the value of your control. It should match exactly the value in the metadata.mat file. Ie if you’re grouping by genotype and your control is the empty split, your control genotype might be ‘emptySplit\_JFRC100\_JFRC49’
   2. Note: This should be enclosed in SINGLE quotes.
8. Path to protocol:
   1. This should be the path to the protocol folder which holds all the fly results. Note that fly folders should only be two levels down from this folder. IE protocol\_folder -> subfolders -> fly folders. If your system is not organized this way, you can set path to protocol equal to whatever folder is two levels above your fly folders. IE if your system is protocol\_folder -> subfolders 1 -> subfolders 2 -> fly folders, set path to protocol equal to the path to subfolders 1. If you do this, when you run the data analysis you will be prompted to browse to the actual protocol folder so the program can get information from your .g4p file.
9. Genotypes
   1. This is an array of names by which the groups should be labeled. These are intended to be simpler, human readable labels representing the metadata values. If you have set a control group, its label should come first, and the labels should be in the same order as field\_values. If field\_values is empty, it will be generated in the order of the group folders alphabetically. IE if your group folders are named Experiment001 – Experiment005 (each containing fly folders), you should list your genotype labels in that order with the exception of the control coming first.
   2. Note: These strings should be encased in DOUBLE quotes.
10. Save path:
    1. A string indicating the path where you’d like your results saved.
11. Plot norm and unnorm:
    1. Equals 0 or 1. If 1, whatever analysis you’re doing will be done twice, once with unnormalized data and once again with normalized data. You still must pass in a normalization flag to tell the software which normalization you would like done, or you will only get the unnormalized analysis. If it is set to zero, you will only get normalized results if you pass in a normalization flag. If you do not pass in a flag, you will only get unnormalized results.
12. Processed data file:
    1. This is a character vector equal to the name of your processed file (minus the .mat). Note that all your flies should have identically named processed files or they will be skipped.
    2. Note this should be enclosed in SINGLE quotes.
13. 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.
14. Annotation text:
    1. This is the annotation text that will appear on your histograms. Any string you want enclosed in double quotes.
15. 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.
16. 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.
17. 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”];
18. Figure names:
    1. An optional array of strings (enclosed in double quotes) providing a figure name for each figure of timeseries plots. Each datatype gets is own figure. If you are using the default layout of timeseries plots, each figure will have a maximum of 30 subplots.
    2. If you’re plotting two datatypes for example, and each has sixty subplots, your figure names should be [“datatype1”, “datatype1”, “datatype2”, “datatype2”].
19. 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’
20. 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.
21. 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
22. 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
23. 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.
    2. If left empty, the analysis will create default subplot titles combining the condition’s pattern and function (if it exists) names.
    3. If you want your subplots to have no titles, set cond\_name = 0
24. 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.
25. CL\_datatypes:
    1. Like OL\_datatypes, this is a cell array of all datatypes you wish to create closed loop histograms for.
26. 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
27. Cond\_name
    1. This cond\_name variable belongs to the TC plot settings (rather than timeseries plot settings) but it serves the same function. If you want to assign custom subplot titles, create an array just like the timeseries cond\_name array, giving a title to each subplot. Ie cond\_name{fig#}(col, row) = “title”;
    2. Leave this empty if you would like default names to be created.
    3. Set to 0 if you would like no titles.
28. 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.
29. Figure names:
    1. TC\_plot\_settings.figure\_names works exactly like the timeseries figure names.
30. 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.
31. TC\_datatypes
    1. Like the other datatypes variables, this is a cell array of all the datatypes you wish to display as tuning curves.
32. 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.

**Creating your data analysis settings file:**

Now that you have your settings as you want them, you need to create a .mat file which contains all your settings preferences. This will be used to actually run the data analysis. We do it this way because it is likely you will have only a few configurations of settings that you will use over and over again, in which case it is easier to create them once and be done.

In G4\_data\_analysis/support there is a function called ‘create\_settings\_file.’ This function takes in two parameters, the name of your settings file and the path where you would like to save it.

Run this function to create a .mat file at the location you specific. This file will be passed in to run the data analysis. Note that if a .mat file already exists with the name and filepath you specify, it will be replaced.

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(path\_to\_settings\_file, ‘-group’, ‘-tsplot’);

The first input is the path to the settings file which you just created. This will tell the class what specifications to use in the analysis. After this 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 if there are many – it would be easier to create a new settings file. 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 or create any settings files.

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:

Create\_settings\_file(filename, filepath) % If you were creating a new settings file  
da = create\_data\_analysis\_tool(path\_to\_settings, ‘-group’, ‘-hist’, ‘-TSplot’, ‘-tcplot’, ‘-normgroup’);  
da.run\_analysis

This will produce a number of graphs, automatically saving them at the save path you entered, then closing so you don’t end up with a large number of windows to x out of. They will be automatically saved in the following way:

Datatype\_groupNames\_plotType\_#.pdf

**Adding new modules:**

Coming soon

**Appendix: Full list of settings:**

Coming soon