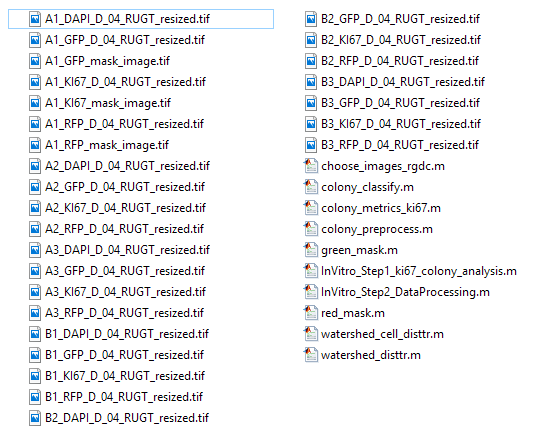
Tutorial

# Introduction.

This tutorial provides an example of how to use the MATLAB scripts and obtain the results as seen in the PLOS one publication entitled A software tool for the quantification of metastatic colony growth dynamics and size distributions *in vitro* and *in vivo.* All functions referred to are custom functions. Functions available in the MATLAB toolbox/library by default are referred to as ‘MATLAB functions’ in this document. Please note the code has been tested on MATLAB 2016a and newer. We cannot guarantee performance on older versions of MATLAB.

# In Vitro Analysis – KI67, RFP, GFP, and DAPI.

Step 1: Place the stitched images of all channels, of all wells in a single folder (i.e. the folder “InVitro\_FixedCell\_Images\_Day\_04” provided at http://pages.jh.edu/~dgilkes1/soumitra/), along with the MATLAB scripts. The content of the folder should appear as below.



Ensure the naming system described in the Readme is followed. In the MATLAB script ‘InVitro\_Step1\_ki67\_colony\_analysis.m’:

1. In the variable ‘all\_wells’, provide the list of wells being analyzed in the six well plate in the variable. The default contains all six wells of the six well plate, as shown in the line –

>>all\_wells = {'A1', 'A2', 'A3', 'B1', 'B2', 'B3'};

1. Specify the threshold arguments in the ‘green\_mask’ and ‘red\_mask’ functions, which are 0.05 and 0.03 in the excerpt below:

>>% RFP and GFP masks

GFP\_mask = green\_mask(GFP, **0.05**);

RFP\_mask = red\_mask(RFP, **0.03**);

The threshold can range from 0 to 1, but we expect that an appropriate value to lie between 0 to 0.15. Following the above code, the script saves the ‘GFP\_mask’ and ‘RFP\_mask’ images as .tif files (the save step is in bold below). For a few wells, these should be compared with the original GFP and RFP images to ensure an appropriate threshold value has been chosen for the experimental set, which leads to creation of an accurate mask in each of the RFP and GFP channels. To compare, go to the folder with the scripts and original images, where the saved GFP and RFP masks will appear, and compare manually.

>> % Save GFP mask as an image

figure, imshow(GFP\_mask);

GFP\_image\_filename = strcat(well, '\_GFP\_mask\_image.tif');

**imwrite(GFP\_mask, GFP\_image\_filename);**

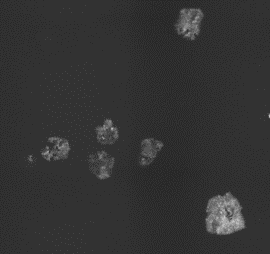
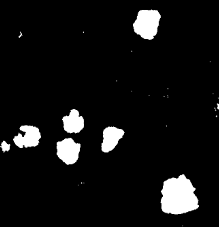
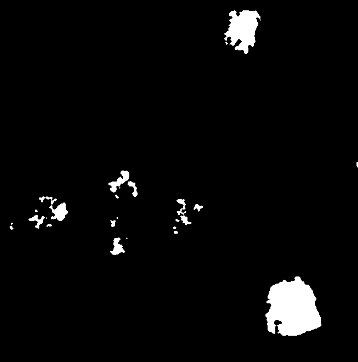
>> % Save RFP mask as an image

figure, imshow(RFP\_mask);

RFP\_image\_filename = strcat(well, '\_RFP\_mask\_image.tif');

**imwrite(RFP\_mask, RFP\_image\_filename);**

An example of accurate and inaccurate segmentation is provided below.

Original Image Accurate mask creation Inaccurate mask creation

Once a threshold is set, it does not need to be changed for a given experiments, or even between experiments if the same cell lines and imaging parameters are used.

1. For the DAPI channel, the threshold can be changed in the function ‘colony\_preprocess.m’ in the section labeled ‘%binarize’. DAPI images should be taken below saturation and should not require threshold changes.
2. The Ki67 threshold is specified in the function ‘colony\_metrics\_ki67.m’ in the section labeled ‘%% KI67 Mask’.

>>%% KI67 mask

>>%figure, imshow(ki67);

>>KI67\_binarized = imbinarize(ki67, **0.3**);

The KI67 mask is saved as an image. The KI67 mask may be compared with the original KI67 channel images by the user.

% Save KI67 mask as an image

%figure, imshow(KI67\_binarized);

KI67\_image\_filename = strcat(well, '\_KI67\_mask\_image.tif');

imwrite(KI67\_binarized, KI67\_image\_filename);

1. The colonies are segmented in the DAPI channel through the ‘watershed\_disttr’ function. This combines the distance transform and watershed segmentation and requires no changes by the user.
2. In line 74 when the ‘colony\_metrics\_ki67’ function is called, change the parameter labeled ‘CountCells’ as ‘0’ for initial runs when establishing accurate segmentation/ threshold values described above in ‘B’.

>>[AreaI, CellCountI, SumInt, SumIntBin, KI67Frac] = colony\_metrics\_ki67(cc\_data, I1, ki67, i, **CountCells**, ShowImages, well)

>>[Area\_i, CellCount\_i, SumInt\_i, SumIntBin\_i, KI67Frac\_i] = colony\_metrics\_ki67(DAPI\_data, DAPI, KI67, i, **1**, 0, well);

Once you are satisfied with the segmentation results (by comparing the binarized images with the originals, as described in Step B), change ‘CountCells’ to ‘1’, which instructs the script to follow begin counting individual nuclei within one colony. This could take several minutes for each well.

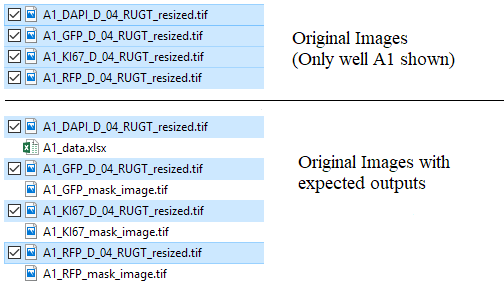
1. The output of this script is a series of excel spreadsheets containing details of each colony as described in the array ‘M’.

>>M(i, :) = [Area\_i, CellCount\_i, SumInt\_i, SumIntBin\_i, KI67Frac\_i, redfrac\_i];

Here, the columns specify area, cell count, sum of intensity, sum of binarized intensity, fraction of nuclei in the colony expressing Ki-67, and ‘Redfrac’ (described in the Readme). The final column specifies ‘Day’. Each row in ‘M’ represents the ‘i-th’ colony in the well. The sum of intensity and sum of binarized intensity of a colony are approximations of the number of nuclei in each colony.

**Expected Outputs:**

1. Mask Images: GFP Mask Image (Step B), RFP Mask Image (Step B), KI67 Mask Image (Step D)
2. Excel Spreadsheet (Step G)



Step 2: Run the second MATLAB script. The input for the second script ‘InVitro\_Step2..’ are the spreadsheets generated in the previous step, at the same location, i.e. the spreadsheets A1\_data.xlxs, A2\_data.xlxs, A3\_data.xlxs, B1\_data.xls, B2\_data.xls, B3\_data.xls.

1. Provide the value for the variable ‘redfrac\_limit’. A value between 0.4-0.6 should work well. =

>>redfrac\_limit = 0.6;

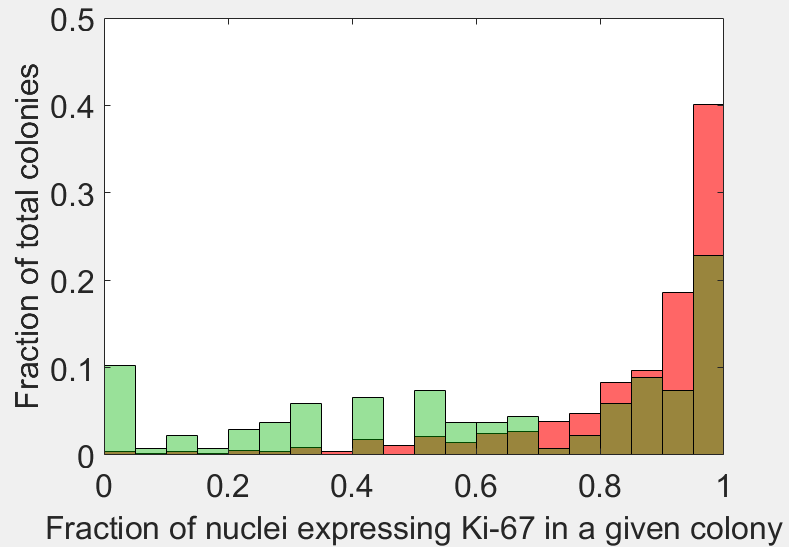
>>well\_data\_temp(well\_data\_temp(:, 6) > 0.6, :) = []; % Remove red colonies, keep green

>>well\_data\_temp(well\_data\_temp(:, 6) < 0.6, :) = []; % Remove green colonies, keep red

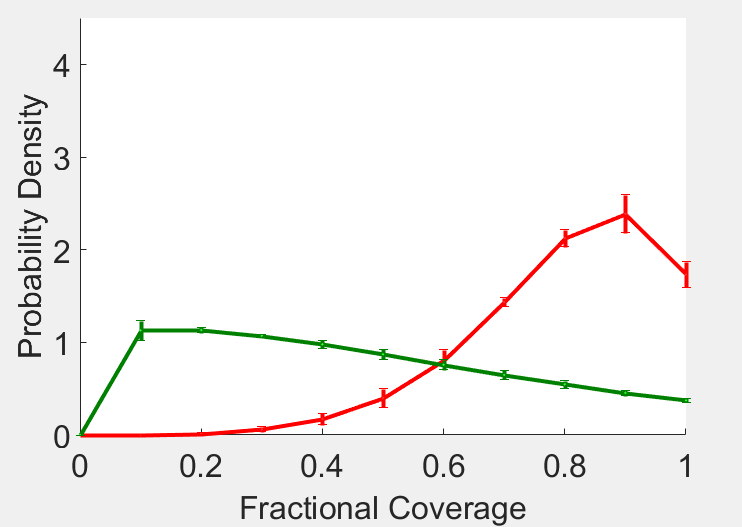
1. The variable ‘all\_wells’ must match the one in Step 1.

>>all\_wells = {'A1', 'A2', 'A3', 'B1', 'B2', 'B3'};

1. The remainder of the script plots the data graphically. The following data is obtained:
2. Ki-67 expression histograms: a comparison between red and green colonies, for each well. There are 6 of such figures, one for each well.



1. An average Ki-67 expression probability density function: Comparison between red and green colonies, averaged across all wells.



The following arrays contain the probability density functions (pdfs) for red and green colonies.

>>% Get Distribution pdf y-values

dist\_red\_y(:,i) = pdf(weibull\_red\_dist, x);

dist\_green\_y(:,i) = pdf(weibull\_green\_dist, x);

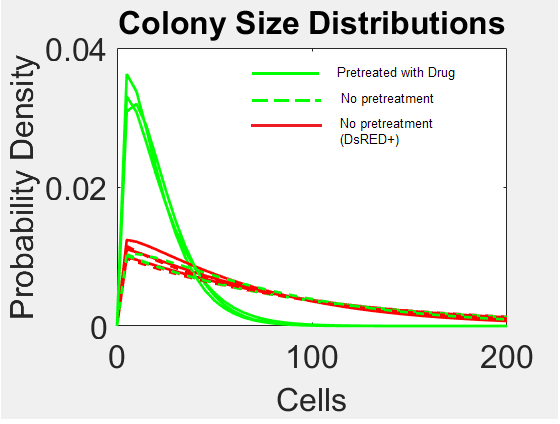
Aside from the above plots, the excel spreadsheets generated from step 1 (see previous step, item ‘G’) can be used if needed to produce custom plots in accordance with the user’s requirements.

1. The variable ‘Normalization’ allows the user to specify whether the data is to be normalized. A value of ‘0’ specifies no normalization, ‘1’ normalizes with respect to the total number of colonies (Red + Green) and ‘2’ normalizes each Red and Green colonies and hence provide a true comparison of the size distributions. ‘
2. The specifics of the figures can be modified in accordance with MATLAB rules and documentation.

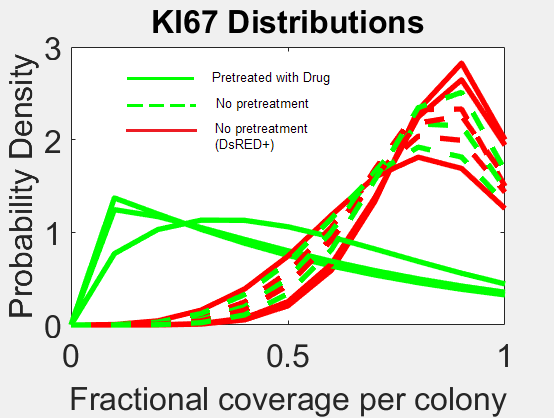
**Expected Outputs:**

MATLAB plots:

1. KI67 expression Histograms (Step C – a.) for each well.
2. KI67 expression average probability function (Step C – b.)
3. A combined plot of colony size distributions (probability density) of each well, below. This serves to check if each set of wells under the same treatment exhibit similar size distributions.

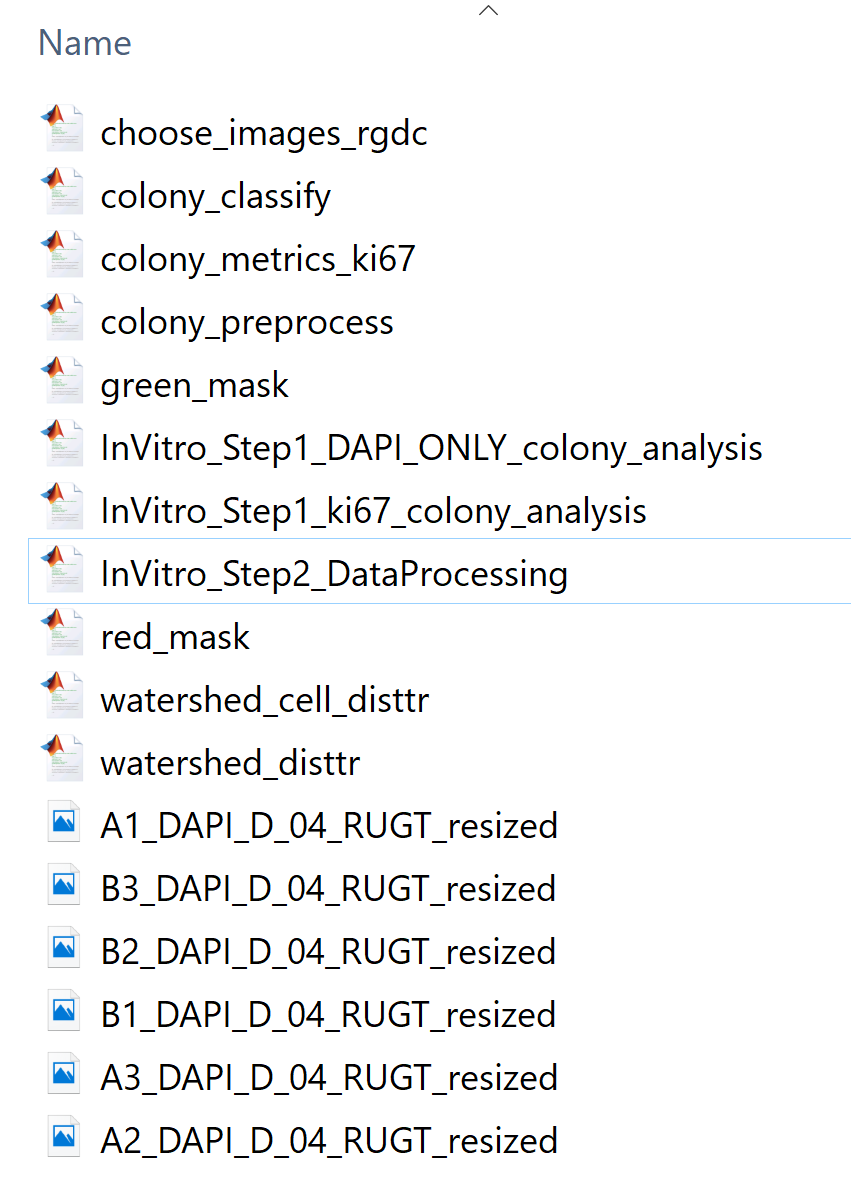


1. Ki-67 Distributions (probability density) of each well, below. This serves to check if each set of wells under the same treatment exhibit similar distributions of tagged protein.



# In Vitro Analysis – DAPI Only

1. Place the stitched images of all wells in a single folder (i.e. the folder “InVitro\_FixedCell\_Images\_Day\_04” provided at http://pages.jh.edu/~dgilkes1/soumitra/), along with the MATLAB scripts. The content of the folder should appear as below.



Ensure the naming system described in the Readme is followed. In the MATLAB script ‘InVitro\_Step1\_DAPI\_ONLY\_colony\_analysis.m’. **NOTE, THIS SCRIPT IS DIFFERENT THAN THE PREVIOUS ONE.**

1. In the variable ‘all\_wells’, provide the list of wells being analyzed in the six well plate in the variable. The default contains all six wells of the six well plate, as shown in the line –

>>all\_wells = {'A1', 'A2', 'A3', 'B1', 'B2', 'B3'};

1. Run the program. Analysis may take several minutes per well as in the previous section. The code will generate an excel spreadsheet for each well. The first column contains the area of the colony, the second contains the number of cells in the colony, the third contains the sum of the intensity of the colony, and the fourth contains binarized sum of intensities.

# In Vitro Analysis – Live Cell.

This section describes using the Live Cell images and scripts to quickly obtain colony area distributions. Place the script ‘Step1\_LiveCellAnalysis.m’, and all the functions, in the same folder as the live cell images hosted at <http://pages.jh.edu/~dgilkes1/soumitra/InVitro_LiveCell_Images/>. Note that the images need to have the format ‘<Well>\_<CHANNEL>\_D\_XX\_<details>.tif’ where ‘XX’ is the day or timepoint where the image was taken. The example shows an experiment with three timepoints – day 1, day 2 and day 4.

The program is written for a 6 well plate where all wells are being used.

Step 1:



1. In the variable ‘all\_wells’, provide the list of wells being analyzed in the six well plate in the variable. It needs to contain all six wells of the six well plate, as shown in the line –

>>all\_wells = {'A1', 'A2', 'A3', 'B1', 'B2', 'B3'};

1. The number of time points is specified in the line below. For our experiment, we have 3 time-points: day 1, day 2 and day 4. The variables ‘i1’ and ‘i2’ (see below) specify the first and the last time-points, allowing the user to choose the ones they wish to analyze. For example, if there are 5 time-points in an experiment, and the user wishes to analyze the 2nd – 4th, ‘i1’ and ‘i2’ must be set to 2 and 4 respectively.

>> % Loop control

i1 = 1; i2 = 3;

% The above variables specify number of time points. The list can be seen

% any time by typing 'list\_r' or 'list\_g' into the console.

1. The preprocessing and binarization is combined in a single function. The threshold must be adjusted by manually comparing segmentation for a few initial images.

>>GFP\_bin = MCF7\_colony\_preprocess\_GFP(I\_green);

>>RFP\_bin = MCF7\_colony\_preprocess\_RFP(I\_red);

The images are saved by the program in the following segment of code:

>> % Save GFP mask as an image

GFP\_image\_filename = strcat(well, '\_GFP\_mask\_image.tif');

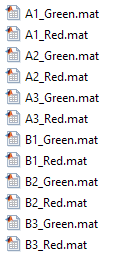
imwrite(GFP\_bin, GFP\_image\_filename);

% Save RFP mask as an image

RFP\_image\_filename = strcat(well, '\_RFP\_mask\_image.tif');

imwrite(RFP\_bin, RFP\_image\_filename);

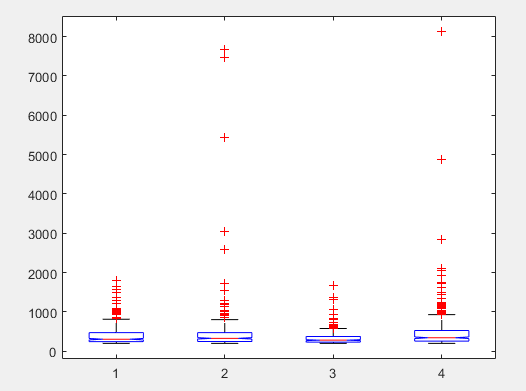
D. The output of the code are two .mat files containing the areas of colonies present in each well:



These .mat files form the input for Step 2 – ‘Step2\_LiveCellAnalysis.m’. The MAT files for all six wells are required for this comparison to run properly.

E. Run the script ‘Step2\_LiveCellAnalysis.m’ to obtain comparisons between Red and green colonies, and between Row A and Row B. The Kruskal-wallis test is used to compare differences in median colony size. More information about the test can be found using the MATLAB documentation, and in the comments provided in the code.

F. The script generates a box and whisker plot where the groups are denoted as follows. Group 1 is red colonies in the first row (Wells A1, A2, A3). Group 2 is red colonies in the second row (Wells B1, B2, B3). Group 3 is green colonies in the first row. Group 4 is green colonies in the 2nd row. These groups are for the Kruskal Wallis test to compare. In this setup, it is assumed that the first row and the second row have different treatments (Ie, no drug in wells A1, A2, and A3 versus drug in B1, B2, and B3)

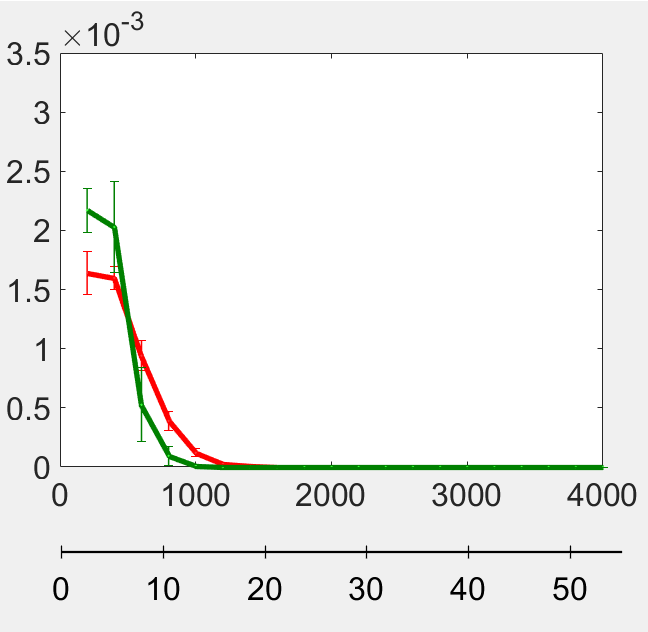


G. Mean probability density is also plotted, as shown in Figure 5 of the paper, under the section ‘Mean Distributions’.

>> %% Mean Distributions  
 % This section plots the mean probability density functions, averaged across each row, for the red and green colonies.

% Plot Drug Treated Wells

distplot\_red = errorbar(dist\_X, mean\_red\_pdf, sd\_red\_pdf, 'r'); hold on  
distplot\_green = errorbar(dist\_X, mean\_green\_pdf, sd\_green\_pdf, 'g');



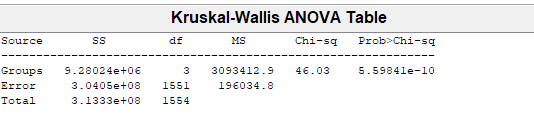
**Expected Outputs:**

Files:

1. .mat files containing well-level data (Step D).

MATLAB Figures

1. Kruskal-Wallis one way ANOVA table: results of the Kruskal-Wallis test.

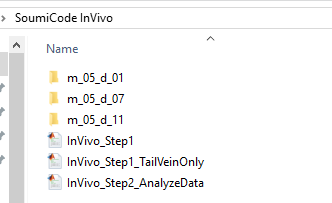


1. Box-and-whisker plot of the results of the Kruskal-Wallis test (Step F).
2. Mean colony size distributions (probability density), averaged across the wells, of the red and green cell lines (Step G).

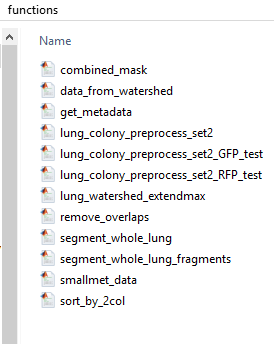
# In Vivo Analysis.

Step 1. Set up the documents in accordance to the instructions mentioned in the readme. One main folder should contain both scripts in addition to subfolders containing channel images for each lung to be analyzed. Subfolders should be named in the convention described in the readme. In a new, separate folder from the main folder, place all of the functions and ensure MATLAB can access this file by selecting and saving it with the "Set Path" button under the Home tab. An example setup for the images in the InVivo\_TailVein Images\_Set\_1 folder on the image hosting website is shown below.

The main folder:



The additional folder where “Set Path” should be set:



The location on the website where these example images are hosted and available for download:



1. Open the script “InVivo\_Step1\_TailVeinOnly.m”. We will begin with this script and attempt to make GFP and RFP masks which match the actual images, similar to part B in the In Vitro step 1. To establish these masks, a preprocessing step must occur. The preprocessing step utilizes the function ‘lung\_colony\_preprocess\_set2\_GFP\_test’ or ‘…RFP…’ as the case may be.

>>[GFP\_adj, GFP\_bin] = lung\_colony\_preprocess\_set2\_GFP\_test(GFP);

>>[RFP\_adj, RFP\_bin] = lung\_colony\_preprocess\_set2\_RFP\_test(RFP);

In the above functions, the threshold level may be specified to obtain a well-segmented image, by changing the threshold in the following line:

>>I\_bin = imbinarize(I\_adj, **0.6**);

Note that the above line and threshold are specified in the function ‘lung\_colony\_preprocess\_set2\_GFP\_test.m’ and not the ‘InVivo\_Step1\_TailVeinOnly.m’ script.

To view the results of the image processing steps remove the comment preceeding the ‘imshow’ MATLAB function, also in the ‘lung\_colony\_preprocess\_set2.m’ function.

>>% figure, imshow(GFP\_binarized);

A few manual iterations of the script should give the user a good estimate of what values to use for the entire set. This is very similar to the In Vitro Step 1, ‘part B’ mentioned above.

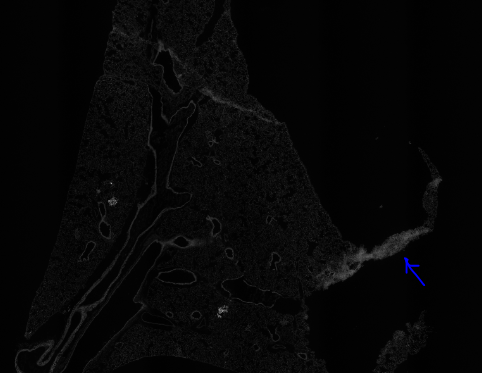
1. As a second alternative, the user may prefer to try the script ‘lung\_colony\_preprocess\_set2.m’ instead. To do this, utilize the script ‘InVivo\_Step1.m’ instead of ‘InVivo\_Step1\_TailVeinOnly.m’.

>> [RFP\_adj, RFP\_bin] = lung\_colony\_preprocess\_set2(RFP);

>> [GFP\_adj, GFP\_bin] = lung\_colony\_preprocess\_set2(GFP);

This function uses Otsu Thresholding and does not require the user to input a threshold value.

1. Note – in some cases, there may be images with high autofluorescence, possibly due to the presence of blood in the fixed tissue. This may impact segmentation accuracy, in which case such sections must be discarded. Alternatively, the area of the tissue with high autofluorescence may be removed manually by the user using a custom script. We provide an example of such a script that can be used as a starting point called ‘Remove\_False\_Areas.m’ which can be found at at http://pages.jh.edu/~dgilkes1/soumitra/InVivo\_TailVein%20Images\_Set\_1/m\_05\_d\_01/ . An example of autofluorescence affecting segmentation in the RFP channel is shown in the image below.



1. In ‘Part 4’, the function ‘combined\_mask’ contains the value that defines the thickness of the boundary (default = 40 pixels). This means that small mets closer than 40 pixels to the nearest ‘large’ met will not be counted, as described in the paper. This boundary thickness can be set manually.

>>% Mask mets already segmented

[GFP\_smallmets, RFP\_smallmets] = ...

combined\_mask( ...

GFP\_watershed, RFP\_watershed, ...

GFP, RFP, ...

DAPI\_lung, **40** ...

);

1. The output is a .xlsx file containing details of RFP and GFP mets. Column one contains the time point in days, column two contains the mouse number, and column three contains met sizes in pixels. This document is the input for the subsequent step. The document represents the bulk of the work, and the user can easily use this output file to create their own custom plots.

**Expected Outputs:**

1. ‘Met\_Areas.xlsx’, as described above in part E.

Step 2.

1. The variable ‘Normalization’ needs to be specified in the same manner as in the In Vitro analysis. Specify the normalization in the for loop with counter ‘**k**’ to produce appropriate figures. ‘1’ normalizes the sum of RFP mets and GFP mets to 1. ‘2’ is used to obtain a true comparison between distributions, and GFP mets and RFP mets are individually normalized to 1.

>> % Choose from 0, 1 or 2.

% 0: No Normalization

% 1: (R + G) = 1

% 2: R = 1 and G = 1

for k = **2:2**

Normalization = k;

1. Metastases consisting of fewer cells than the value specified in ‘MinCellLimit’ are ignored. Also, histograms are plotted up to the largest tumor (X-axis), specified in the variable ‘max\_tumor\_area’.

>> max\_tumor\_area = 100000;

>> edges = linspace(0,max\_tumor\_area/cell\_area\_in\_pixels, 100);

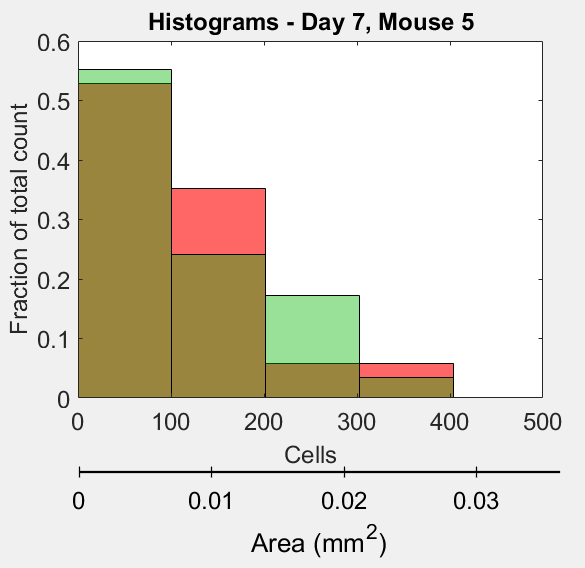
>> MinCellLimit = 3;

1. The subsequent part of the script represents the data graphically, based on the Normalization option chosen. The results of the script are shown in Figure 7 of the paper. More details are provided in the form of comments in the MATLAB script.
2. The graph-making scripts use MATLAB functions which are well documented online. The MATLAB function ‘ranksum’ is used to evaluate the Wilcoxon Rank-Sum test.

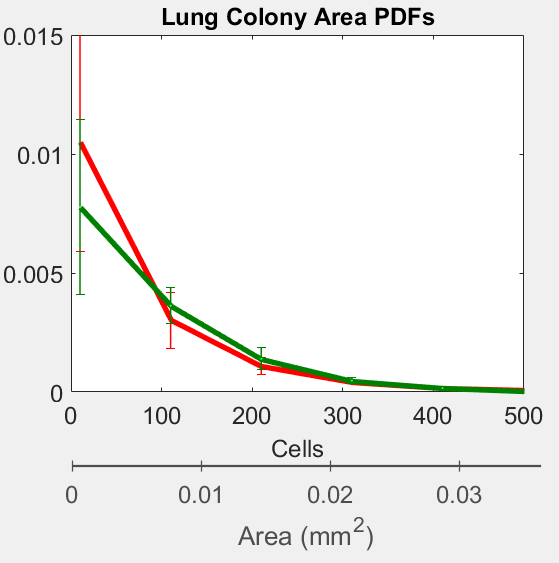
**Expected Outputs:**

Matlab Figures

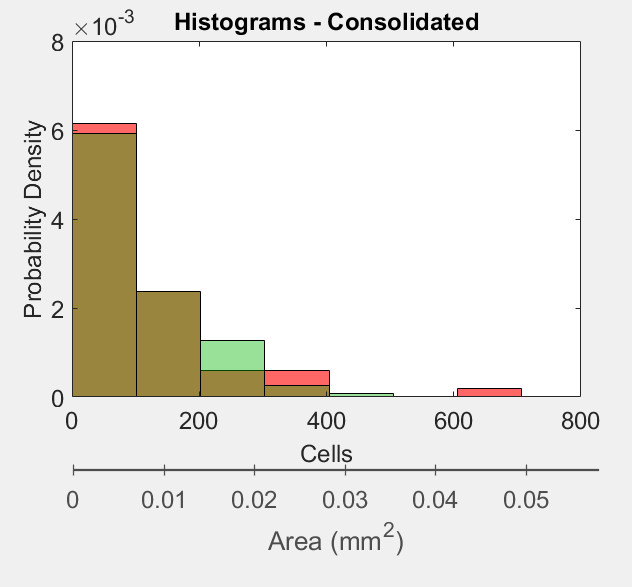
1. Colony size distributions of individual lung sections (below), for each lung section analyzed, also shown in Figure 7 D of the manuscript.



1. Colony size distributions averaged over all lung sections (below, and in Figure 7C of the manuscript).



1. Colony size distribution of lung sections consolidated (below). Metastatic colonies from all the lung sections used are grouped into a single set and binned.



1. Comparing colony size distributions of two different lung sections (below). If the user wishes to plot a comparison of the colonies of any two sections of their choice, they may do so in the code below the following annotation:

>>%% Fit Individual distributions:

% This graph compares the colony size distributions between two lungs.

% The user can manually select the lungs they wish to compare below.

% The numbers are the order of lungs in the variables 'RFP\_data' and

% 'GFP\_data' (type them in the console and press enter to see the 'day'

% and 'mouse number'.

The dashed lines represent the red and green colonies in one lung section, and the solid lines represent the second lung section so chosen.

