This tutorial explains how to perform a batch analysis given a *Color Model*. If users have not created a *Color Model (CM)* yet, then the corresponding tutorial can be accessed here.

Performing a Batch Analysis

As a starting point for users, a MATLAB script with a template batch analysis is available here. The script is labeled as *batch_template.m*. Users should proceed to download and open this file using MATLAB or Octave.

The first step within this script is to load the *Color Model(s)* of the associated Flow Cytometry data to measure. To do this, we upload the *.mat* file for the *CM* using the **load**. Then the meta-data such as the experiment name of what this analysis measures can be set.

```
1 load('../template_colormodel/CM120312.mat');
2 experimentName = 'LacI Transfer Curve';
```

Once the data is loaded, we can begin to configure the analysis. To set the x_axis scale for the data, we create an instance of a **BinSequence**. For example, *line 1* only analyzes data on a histogram scale of 10^(first) to 10^(third) MEFL, measuring the bins every 10^(second).

The first parameter is the minimum data values considered. The second parameter sets the bin step. The third parameter sets the maximum data values considered. Lastly, the final parameter identifies the mode of how the data should be evaluated. There are three main mode choices: *"geometric"* which centers the data at geometric mean of the edges, *"log_bins"* which sorts the bins geometrically, but the input values are on a log10 scale, and *"arithmetic"* which centers the data at arithmetic mean of the edges.

The next step is identifying a few more environment settings. An instance of the **AnalysisParameters** needs to be set which designate which channels have which roles and apply the bins as shown on **line 2**. Then to organize the data within the bins more, we set **setMinValidCount**. This function ignores any bins with less than the valid count set as the second parameter to be noise. The function **setPemDropThreshold** ignores any raw fluorescence values less than this threshold as too contaminated by instrument noise. Then, using **setUseAutoFluorescence**, we can add autofluorescence back in after removing it previously to account for compensation.

```
1 bins = BinSequence(4,0.1,10,'log_bins');
2 AP = AnalysisParameters(bins,{});
3 AP=setMinValidCount(AP,100');
4 AP=setPemDropThreshold(AP,5');
5 AP=setUseAutoFluorescence(AP,false');
```

Once the data settings are identified, then we need to map the file sets to their condition names as shown below. Currently, the script is using the example assays provided as the pathway to the files, users need to change this to point to their data.

Once the files are set up, the actual analysis can be run using the **per_color_consitutive_analysis** function which identifies the colors analyzed. Currently, the script analyzes: *EBFP2*, *EYFP*, and *mKate*. Users can change the color channels to include more or less.

```
% Make a map of condition names to file sets
stem1011 = '../example_assay/LacI-CAGop_';
file_pairs = {...
  % Replicates go here, e.g., {[rep1], [rep2], [rep3]}
  'Dox 0.1',    {[stem1011 'B3_B03_P3.fcs']};
  'Dox 0.2',    {[stem1011 'B4_B04_P3.fcs']};
  'Dox 0.5',    {[stem1011 'B5_B05_P3.fcs']};
  'Dox 1.0',    {[stem1011 'B6_B06_P3.fcs']};
  'Dox 2.0',    {[stem1011 'B7_B07_P3.fcs']};
  'Dox 5.0',    {[stem1011 'B8_B08_P3.fcs']};
  'Dox 10.0',   {[stem1011 'B9_B09_P3.fcs']};
  'Dox 20.0',   {[stem1011 'B10_B10_P3.fcs']};
  'Dox 50.0',   {[stem1011 'B11_B11_P3.fcs']};
  'Dox 100.0',  {[stem1011 'B12_B12_P3.fcs']};
  'Dox 200.0',  {[stem1011 'C1_C01_P3.fcs']};
  'Dox 500.0',  {[stem1011 'C2_C02_P3.fcs']};
  'Dox 1000.0', {[stem1011 'C3_C03_P3.fcs']};
  'Dox 2000.0', {[stem1011 'C4_C04_P3.fcs']};
  };

n_conditions = size(file_pairs,1);

% Execute the actual analysis
[results sampleresults] = per_color_constitutive_analysis(CM,file_pairs,{'EBFP2','
    EYFP','mKate'},AP);
```

Once the analysis is complete, the last step is to create plots and output the data to *csv* files.

```
% Make output plots
OS = OutputSettings('LacI-CAGop','','','plots');
OS.FixedInputAxis = [1e4 1e10];
plot_batch_histograms(results,sampleresults,OS,{'b','y','r'});

save('LacI-CAGop-batch.mat','AP','bins','file_pairs','OS','results','sampleresults
    ');

% Dump CSV files:
fprintf('Dumping CSV files\n');
fid = fopen('LacI-CAGop-batch.csv','w');
fprintf(fid,'Device ID,datapoints,,,log10 Mean,,,Std.Dev. of mean (fold)\n');
fprintf(fid,',EBFP2,EYFP,mKate,EBFP2,EYFP,mKate,EBFP2,EYFP,mKate\n');
for i=1:n_conditions
    fprintf(fid,'%s,',file_pairs{i,1});
    fprintf(fid,'%d,',sum(results{i}.bincounts));
    fprintf(fid,'%d,',log10(results{i}.means));
    fprintf(fid,'%d,',results{i}.stdofmeans);
    fprintf(fid,'\n');
end
fclose(fid);
```