Basic tips for creating visually appealing graphics in MATLAB®, from the Golding Lab

Contents

- Constructing a simple plot.
- Beautifying a plot.
- Recap: basic parameters to control the appearance of a plot.
- Choosing colors. Option I: picking colors manually.
- Choosing colors. Option II: Using pre-defined colormaps.
- Choosing colors. Option III: Creating your own colormaps.
- Displaying multiple datasets on a plot.

MATLAB® is a software and programming environment that allows the manipulation and visualization of data. Using a simple scripting language, one can combine numerical computations with complex graphical displays, which makes MATLAB® a very useful tool for data analysis and exploration. However, despite all its versatility, the default graphic options from MATLAB® are not very appealing, either for data exploration or further publication. Below, I will give a series of simple tips to improve graphics in MATLAB®.

Constructing a simple plot.

We will start by creating a very simple data set.

```matlab
% Close previous plots.
close all; % Close previous plots.
clear; % Delete all variables from the workspace.
clc; % Delete any past commands from the command line.
N = 100;
X = 1:N;
Y = 1:N;
```

The code above generates two equal numeric arrays, \( X \) and \( Y \), that contain \( N \) integer numbers (from 1 to \( N \)). Then, we can add some noise to the data:

```matlab
X = X + 0.5*N*(rand(1,N)-0.5); % rand(A,B) generates an array with A rows and B columns filled with random numbers uniformly distributed between 0 and 1.
Y = Y + 0.5*N*(rand(1,N)-0.5);
```

The code above adds a random number, ranging from -0.25*\( N \) to 0.25*\( N \), to each element of \( X \) and \( Y \). We can display the data using the `plot` function:

```matlab
fh = figure; % Open a new figure, and creates the variable fh that contains all the properties of the figure.
hold on; % Hold the current plot and all axis properties so that subsequent graphing commands add to the existing graph.
plot(X,Y,'o'); % Plots X vs Y, using circles 'o' as markers.
```
Now, let's calculate the mean and standard error over finite windows of the data:

```matlab
n_bins = 20;
x_mean = zeros(n_bins,1); % zeros(A,B) generates an array with A rows and B columns filled with zeros.
x_sem  = zeros(n_bins,1);
y_mean = zeros(n_bins,1);
y_sem  = zeros(n_bins,1);
for ii = 1:n_bins
    % Select the index of the data sets to be merged into one bin.
    bin_idx    = (1:N/n_bins) + (ii-1)*N/n_bins;
    % Mean X value in the bin.
    x_mean(ii) = mean(X(bin_idx)); % mean(A): average or mean value of A.
    % Standard Error of the Mean (SEM).
    x_sem(ii)  = std(X(bin_idx))/sqrt(numel(bin_idx)); % sqrt(A): square root of A.
    % numel(A): number of elements of array A. (a simple variable is considered an 1x1 array)
    % Mean Y value in the bin.
    y_mean(ii) = mean(Y(bin_idx));
    % Standard Error of the Mean (SEM).
    y_sem(ii)  = std(Y(bin_idx))/sqrt(numel(bin_idx));
end

We can now overlap the average data plus error bars on top of the raw data:

```errorbar(x_mean,y_mean,y_sem,'ro');```
MATLAB® has an extensive *Curve Fitting Toolbox™*. Another feature of a simple plot is to add a simple model trying to describe the data. For illustration, we can try to fit the raw data with a linear model:

```matlab
% set fitting parameters
ft = fittype( 'poly1' ); % Define the type of function used to fit. In this case, a polynomial of grade 1, or y = p1*X + p2.
opts = fitoptions( ft ); % Generate structure array with parameters for the fit.
opts.Lower = [-Inf -Inf]; % Lower bounds for p1 and p2 parameters.
opts.Upper = [Inf Inf]; % Upper bounds for p1 and p2 parameters.

% Fit model to data.
[fitresult, gof] = fit( X', Y', ft, opts );
```

And we can now add the fit to the plot:

```matlab
xfit = -0.1*N:1.1*N;
yfit = fitresult.p1*xfit + fitresult.p2;

% Calculate the standard error in the parameters.
ci   = confint(fitresult,0.95); % Get the 95% confidence intervals (CI) for the fit.
p1_se = (ci{2,1}-ci{1,1})/4; % CI is approximately 2*SEM.
p2_se = (ci{1,2}-ci{2,2})/4; % for more info see https://en.wikipedia.org/wiki/Confidence_interval.

plot(xfit,yfit,'k-');
```
Beautifying a plot.

One of the main concepts in graphical style is Salience. Salience is *a visual quality that sets an object apart from its surroundings* (Bang Wong, Nature Methods 2010). The salience of an object can be modified by changing attributes such as color, thickness (of lines) and size. The plot above, made using some of the default colors available in MATLAB®, illustrates why a poor color selection needs to be avoided: it not only makes things look ugly, but most importantly, makes graphics hard to understand.

Below is an improved version of the figure above, were we changed the color, thickness and size of the markers. In the original plot, the three groups of objects on display (the raw data, the moving average and the fit) had saturated colors that are very hard to distinguish from each other. One can improve the plot by decreasing the salience of the raw data (in the case below, by using a light gray color) and then keeping the colors of the average data and the fit:

```matlab
fh = figure; hold on
set(fh,'color','w'); % Define the background color of the figure. The default is gray.

% Plot data and set different object properties, like, MarkerSize
% MarkerFaceColor, MarkerEdgeColor and LineWidth. plh, elh and p2h are
% handles to the plots that can be used later to modify the plots further.

plh = plot(X,Y,'o','MarkerSize',5,...
'MarkerFaceColor',[.85 .85 .85],... % 'MarkerEdgeColor','none');

elh = errorbar(x_mean,y_mean,y_sem,'ko',...
'MarkerSize',7,...
'MarkerFaceColor','k',...
'MarkerEdgeColor','none',...
'LineWidth',2);

p2h = plot(xfit,yfit,'r-',...
'LineWidth',2);
```
I would argue that the new, "beautified" plot is clearer than the original, unformatted plot. The changes made are listed below:

- We increased the width of the fit and average lines to 2 pt. This increase the salience of the data with respect to the figure axis (which remains to be 0.5 pt).
- We decreased the salience of the raw data with respect of the average data by increasing the brightness of the color.

After improving the general appearance of the plot, we need to add some comments to help make sense of the data. The most typical annotations are the labels of the X and Y axis, as well as adding a brief explanatory title:

```matlab
fontsize = 13;
xlabel('X variable (A.U.)','fontsize',fontsize);
ylabel('Y variable (A.U.)','fontsize',fontsize);
title(['X and Y are linearly correlated (\rho = ' num2str(corr(X',Y'),'%1.3f') ' )'],'fontsize',fontsize);
```
Given that both axes have the same numerical range, we can adjust the axes aspect-ratio:

```matlab
axis square;  % Adjust the axes to have the same size in the X and Y dimensions.
```

Additionally, we can adjust the limits of the axes:

```matlab
xlim([-0.1 1.1]*N);  % xlim([MIN MAX]);
ylim([-0.1 1.1]*N);
```
Another important concept of graphical style is **Simplicity**. If possible, it is important to remove unnecessary elements from a figure. In the case above, each axis has many tick marks. We can adjust which ticks and labels to show, and evaluate whether the figure has improved:

```matlab
% gca: get handle of the current axis.
set(gca,'XTick',[0 N/2 N],...
    'XTickLabel',{'0',num2str(N/2),num2str(N)},...
    'YTick',[0 N/2 N],...
    'YTickLabel',{'0',num2str(N/2),num2str(N)},...
    'fontsize',12);
```

We can also try closing the bounding box of the plot:
Finally, we can add an explanatory legend describing what is being plotted.

```matlab
lh = legend([p1h e1h p2h], 'raw data',... 'moving average',... ['Y = p_1*X + p_2' sprintf('
') ... 'p_1 = ' num2str(fitresult.p1,'%1.2f') '\pm'  num2str(p1_se,'%1.2f') ... sprintf('
')... 'p_2 = ' num2str(fitresult.p2,'%1.2f') '\pm'  num2str(p2_se,'%1.2f')]); set(lh,'Location','NorthWest','Fontsize',10); % Set the properties of the legend.
```
Recap: basic parameters to control the appearance of a plot.

Below is a short script that includes all the options that we used to improve the appearance of our plot. It can be adapted to fit your plotting needs:

```matlab
% 1. Set the figure properties
fh = figure;
hold on;
set(fh, 'color', 'w',...
    'units', 'inches',...
    'position', [.2 .5 9 9]);
% Open figure and create legend handle fh.
% Set background color of the outside of figure. gray is default. White (w) looks better.
% Position of the figure on the screen.
% Distance to left of screen ...
% Distance to bottom of screen ...
% Width of the figure ...
% Height of the figure

% 2. Set the properties of the plot
plh = plot(rand(100,1),rand(100,1), 'o', ...
    'MarkerSize', 15, ...
    'MarkerFaceColor', [.85 .85 .85], ...
    'MarkerEdgeColor', [0.88 0.28 0.44], ...
    'LineWidth', 3);
% Possible markers = o,s,d,v,^,.,*.
% Size of marker.
% Color of the marker. 'None' is the default.
% Color of the edge line of the marker. 'None' to not show.

% 3. Set the properties of the plot axis, add annotations
% Below are the most common options (the ones I use more often)
% for a more exhaustive list, see:
% http://www.mathworks.com/help/matlab/ref/axes-properties.html
set(gca, ...
    'XTick', [0 .25 .50 .75 1],...
    'XTickLabel', {'0','.25','.50','.75','1'},...
    'XTickLabel', [0.18 0.56 1.00],...
    'XColor', [0.18 0.56 1.00],...
    'XDir', 'normal',...
    'YTick', [0 .50 1],...
    'YTickLabel', {'0','.50','1'},...
    'YColor', [0.20 0.80 0.20],...
    'YDir', 'reverse',...
    'FontSize', 20, ...
    'Linewidth', 2);
% List of positions for the X ticks.
% List of labels for the X ticks.
% Range of the of X axis on display.
% Direction of the axis. 'normal' is ascendent.
% List of positions for the Y ticks.
% List of labels for the Y ticks.
% Range of the of Y axis on display.
% Direction of the Y axis. 'reverse' is descendent.
% Font size of the Xtick and Y tick labels.
% Width of the axes and ticks. 0.5 is the default.

axis square;
off to not show axis.
xlabel('X label','fontsize',24);
ylabel('Y label','fontsize',24);
title('Title','fontsize',30);
box on;
% Define axes aspect ratio and visibility. Use 'axis
% Add Explanatory label to X axis.
% Add Explanatory label to Y axis.
% Add explanatory Title to the figure.
% Add (on) or remove (off) the lines on the top and left
% of the axis.

% 4. Add a legend and set its properties
lh = legend(plh, 'raw data');
set(lh, 'Location', 'NorthWest', ...
    'FontSize', 25);
% Add explanatory legend.
% Define location of the legend in the axis frame of reference.
% Legend font size.
```
Choosing colors. Option I: picking colors manually.

Color is an important tool to modify the salience of objects. There are multiple ways to choose colors. The first and simplest, is to choose manually from a color picker inside a graphical program like illustrator. I like to visit this website: [http://cloford.com/resources/colours/500col.htm](http://cloford.com/resources/colours/500col.htm), choose some colors that I find appealing, and try to modify them later in illustrator. Below, I created an array with the RGB codes for all the colors from the website (without the gray and black colors) to display them inside MATLAB®:

```
C = ...  
```
I display the colors below, using the plot function and a for loop:

```matlab
fh = figure; hold on;
set(fh,'color','w')
for ii = 1:20
    for jj = 1:20
        counter = counter + 1;
        plot(ii,jj,'s','markersize',14,'markeredgecolor','none',...)
        counter = counter + 1;
    end
end
```
Choosing colors. Option II: Using pre-defined colormaps.

Other good source of colors is the website http://colorbrewer2.org/. One can download the MATLAB® version from the MathWorks File Exchange: https://www.mathworks.com/matlabcentral/fileexchange/34087-cbrewer---colorbrewer-schemes-for-matlab. Once you download the program to your computer, you can add the folder containing the code to the matlab path:

cbrewer_folder = [ pwd 'cbrewer']; % pwd gets the path to the current folder.
addpath(cbrewer_folder);

Warning: Name is nonexistent or not a directory: C:\Users\laduran\Desktop\research\presentations\2016_06_07 Figure making\cbrewer.

If we call the cbrewer command, it will display the available colormaps:

cbrewer;

[colormap] = cbrewer(ctype, cname, ncol [, interp_method])

INPUT:
- ctype: type of color table *seq* (sequential), *div* (divergent), *qual* (qualitative)
- cname: name of colortable. It changes depending on ctype.
- ncol: number of color in the table. It changes according to ctype and cname
- interp_method: interpolation method (see interpl.m). Default is "cubic"
Basic tips for creating visually appealing graphics in MATLAB®, from the Golding Lab

'C' 'Blues' 'BuGn' 'BuPu' 'GnBu' 'Greys' 'Oranges' 'OrRd' 'PuBu' 'PuBuGn' 'PuRd' 'Purples' 'RdPu' 'Greens' 'YlGn' 'YlGnBu' 'YlOrBr' 'YlOrRd'

Divergent tables:
- 'BrBG'
- 'PiYG'
- 'PRGn'
- 'PuOr'
- 'RdBu'
- 'RdGy'
- 'RdYlBu'
- 'RdYlGn'

Qualitative tables:
- 'Accent'
- 'Dark2'
- 'Paired'
- 'Pastel1'
- 'Pastel2'
- 'Set1'
- 'Set2'
- 'Set3'

Cbrewer provides three types of colormaps. Divergent and Sequential colormaps are better to display the continuous change in a quantity. In contrast, the Qualitative colormaps are intended to provide colors to separate data into groups. Below, I compare the default MATLAB colormap (called 'Jet') with one of the diverging colormaps available from the cbrewer function (called 'Spectral'):

```matlab
fh = figure;
```
The Jet colormap is made of strong, saturated colors. Saturated colors are easy to see, but may overwhelm the reader. A colormap with less saturated colors, like the Spectral colormap
Choosing colors. Option III: Creating your own colormaps.

Alternatively, you can create a custom colormap. Bang Wong, talking about color coding in his column *Points of View*, suggest a simple way for creating qualitative colormaps using the Hue/Saturation/Lightness colorspace (HSL). Below is an excerpt from the column, describing HSL:

*Every color is described by three properties: hue, saturation and lightness. Hue is the attribute we use to classify a color as red or yellow. Saturation describes the neutrality of a color; a red object with little or no white is said to be very saturated. The lightness of a color tells us about its relative ordering on the dark-to-light scale.*

(Excerpt from http://www.nature.com/nmeth/journal/v7/n8/full/nmeth0810-573.html)

Below, I create a set of Qualitative colormaps by creating colors in the HSV colorspace (very similar to HSL) and then converting the colors back to RGB.

```matlab
figure('color','w'); hold on;

% Numbers of colormaps to create.
cmapNum = 10;
% Number of colors per colormap.
colorNum = 10;

rgb_color = zeros(colorNum,3,cmapNum);

for jj = 1:cmapNum
    % Define vectors where we simultaneously change Hue, Saturation and Brightness.
    Brightness = 0.95:-(0.95-0.5)/colorNum:0.5;  % From 0.5 to 0.95.
    Saturation = 0.1:0.9/colorsNum:.9;            % From 0.1 to 0.9.
    Hue = 0:1/colorNum:

    for ii = 1:colorNum
        % Define a color in the hsv scale and convert to rgb.
        rgb_color(ii,:,jj) = hsv2rgb([mod(Hue(ii) + jj/cmapNum,1) Saturation(ii) Brightness(ii) ]); % Hue Value/Saturation/Brightness.

        plot(ii,2*jj,'s','markerfacecolor',squeeze(rgb_color(ii,:,jj)),... % Squeeze remove unnecessary dimensions of an array.
             'markeredgecolor','none',...
             'markersize',20);
    end
end

axis equal;
xlim([0 colorNum*1.1]);
ylim([0 2*cmapNum]);
set(gca,'Xtick',[],'XColor',[1 1 1],'YColor',[1 1 1]);
xlabel('Decreasing Brightness (Value)');
ylabel('Increasing Saturation');
annotation('textarrow',[.33 .7],[0.15 0.15]);
```

(Taken from Figure_guide_LS_2016.htm)
Basic tips for creating visually appealing graphics in MATLAB®, from the Golding Lab

Above we changed the color of the axis to white instead of using 'axis off', because the latter would have deleted also the X label. We also used a nifty annotation function to create an arrow. We can now test whether the colormap is able to differentiate datasets in a plot:

```matlab
figure('color','w'); hold on;
cmap_idx = 5; % Choose one of the colormaps.
for ii = 1:colorNum
    % Create ColorNum random datasets, with 20 elements each.
    thetal1 = 2*pi*rand(20,1); % Generate 20 random angles between 0 and 2*pi.
    radius1 = rand(20,1); % Generate 20 random radii between 0 and 1.
    x1 = radius1.*cos(thetal1); % Transform to cartesian coordinates.
    y1 = radius1.*sin(thetal1);
    thetal2 = 2*pi*rand(1); % Generate 20 random angles between 0 and 2*pi.
    radius2 = 5*rand(1); % Generate 20 random radii between 0 and 5.
    x2 = radius2.*cos(thetal2); % Transform to cartesian coordinates.
    y2 = radius2.*sin(thetal2);
    x = x1+x2; % Generate data inside an unity circle that is inside a circle of radius 5.
    y = y1+y2;
    plot(x,y,'o','markerfacecolor',squeeze(rgb_color(ii,:,cmap_idx)),...
         'markeredgecolor','none',...
         'markersize',8);
end
axis equal;
xlim([-5 5]);
ylim([-5 5]);
box on;
set(gca, 'XTick', [-5 0 5], 'XTickLabel', {'-5', '0', '5'},...
     'YTick', [-5 0 5], 'YTickLabel', {'-5', '0', '5'});
title('Qualitative colormap, {\itala} Bang Wong');
```
Displaying multiple datasets on a plot.

Above I showed a single colormap example. But many times, when we analyze scientific data, it is useful to show multiple plots in a figure, to help us compare different data sets. MATLAB® has a simple function, called `subplot`, that allows us to allocate multiple plots into one figure. Below, I show all the possible colormaps created in above in an array of 3x3 plots:

```matlab
figure('color','w'); hold on;
for jj = 1:9 % For each colormap.
    subplot(3,3,jj); hold on;
    for ii = 1:colorNum
        % Create ColorNum random datasets, with 20 elements each.
        theta1 = 2*pi*rand(20,1);          % Generate 20 random angles between 0 and 2*pi.
        radius1 = rand(20,1);               % Generate 20 random radii between 0 and 1.
        x1     = radius1.*cos(theta1);     % Transform to cartesian coordinates.
        y1     = radius1.*sin(theta1);
        theta2 = 2*pi*rand(1);             % Generate 20 random angles between 0 and 2*pi.
        radius2 = 5*rand(1);                % Generate 20 random radii between 0 and 5.
        x2     = radius2.*cos(theta2);     % Transform to cartesian coordinates.
        y2     = radius2.*sin(theta2);
        x      = x1+x2;                    % Generate data inside an unity circle that is inside a circle
        y      = y1+y2;

        plot(x,y,'o','markerfacecolor',squeeze(rgb_color(ii,:,jj)),...
             'markeredgecolor','none',...
             'markersize',5);
    end
    axis equal;
    xlim([-5 5]);
    ylim([-5 5]);
    box on;
    xlabel('X');
    ylabel('Y');
    set(gca,'Xtick',[-5 0 5],'XTickLabel',{'-5','0','5'},...
         'Ytick',[-5 0 5],'YTickLabel',{'-5','0','5'});
end
```
To visualize the plots better, you can increase the size of the figure:

```matlab
figure('color','w','units','inches','position',[.2 .5 9 9]); hold on;
for jj = 1:9 % For each colormap.
    subplot(3,3,jj); hold on;
    for ii = 1:colorNum
        % Create ColorNum random datasets, with 20 elements each.
        theta1  = 2*pi*rand(20,1);          % Generate 20 random angles between 0 and 2*pi.
        radius1 = rand(20,1);               % Generate 20 random radii between 0 and 1.
        x1      = radius1.*cos(theta1);     % Transform to cartesian coordinates.
        y1      = radius1.*sin(theta1);
        theta2  = 2*pi*rand(1);             % Generate 20 random angles between 0 and 2*pi.
        radius2 = 5*rand(1);                % Generate 20 random radii between 0 and 5.
        x2      = radius2.*cos(theta2);     % Transform to cartesian coordinates.
        y2      = radius2.*sin(theta2);
        x       = x1+x2;                    % generate data inside an unity circle that is inside a circle
        y       = y1+y2;
        plot(x,y,'o','markerfacecolor',squeeze(rgb_color(ii,:,jj)),...  
             'markeredgecolor','none',... 
             'markersize',8);
    end
    axis equal;
    xlim([-5 5]);
    ylim([-5 5]);
    box on;
    xlabel('X');
    ylabel('Y');
    set(gca,'Xtick',[-5 0 5],'XTickLabel',
             {'-5','0','5'},... 
             'Ytick',[-5 0 5],'YTickLabel',
             {'-5','0','5'});
end
```
Below, is an example where I create 100 different colormaps, and display them in a 10x10 subplot array, using a scatter plot of random data as done above:

```matlab
figure('color','w','units','inches','position',[.2 .5 9 9]); hold on;
cmapNum = 100; % Number of colormaps to create.
colorNum = 10; % Number of colors per colormap.
for jj = 1:cmapNum
    % Define vectors where we simultaneously change Hue, Saturation and Brightness.
    Brightness = 0.95:-(0.95-0.5)/colorNum:0.5; % From 0.3 to 0.95.
    Saturation = 0.1:(0.9-0.1)/colorNum:.9; % From 0.1 to 0.9.
```

Figure_guide_LS_2016.htm[1/18/2017 11:13:17 AM]
Hue = 0:1/colorNum; % From 0.1 to 1.0.

subplot(10,10,jj); hold on;

for ii = 1:colorNum

% Define a color in the hsv scale and convert to rgb.
rgb_color = hsv2rgb([ mod(Hue(ii) + jj/cmapNum,1) Saturation(ii) Brightness(ii)]); % Hue Saturation Value/Brightness.

% Create ColorNum random datasets, with 20 elements each.
theta1  = 2*pi*rand(20,1); % Generate 20 random angles between 0 and 2*pi.
radius1 = rand(20,1); % Generate 20 random radii between 0 and 1.
x1      = radius1.*cos(theta1); % Transform to cartesian coordinates.
y1      = radius1.*sin(theta1);

theta2  = 2*pi*rand(1); % Generate 20 random angles between 0 and 2*pi.
radius2 = 5*rand(1); % Generate 20 random radii between 0 and 5.
x2      = radius2.*cos(theta2); % Transform to cartesian coordinates.
y2      = radius2.*sin(theta2);

x       = x1+x2; % generate data inside an unity circle that is inside a circle of radius 5.
y       = y1+y2;

plot(x,y,'o','markerfacecolor',rgb_color,...
     'markeredgecolor','none',... % marker is not visible.
     'markersize',5);
end

axis equal;
xlim([-5 5]);
ylim([-5 5]);
box on;

% Only display Xlabels, Ylabels and ticks in plots located at the bottom and the left of the figure.

fontsize = 10;

if mod(jj,10) == 1 && jj ~= 91
    set(gca,'Xtick',[],...
         'XTickLabel',{-5,'0','5'},...
         'FontSize',fontsize);
ylabel('Y');
elseif jj == 91
    set(gca,'Xtick',[-5 0 5],
         'XTickLabel',{-5,'0','5'},...
         'Ytick',[-5 0 5],
         'YTickLabel',{'-5','0','5'},...
         'FontSize',fontsize);
ylabel('Y');
else jj > 91
    set(gca,'Xtick',[-5 0 5],
         'XTickLabel',{-5,'0','5'},...
         'Ytick',[-5 0 5],
         'YTickLabel',{'-5','0','5'},...
         'FontSize',fontsize);
xlabel('X');
else
    set(gca,'Xtick',[];
         'XTickLabel',{};
         'Ytick',[];
         'YTickLabel',{};
         'FontSize',fontsize);
end
Above, in addition to show multiple plots, we only show the ticks and labels of subplots at the bottom and left of the subplot array, to save some space, and in this way make more room for the figures. However, that only partially helps. To have a greater control of the appearance of subplots, one can use the `subaxis` function instead. The first thing to do is to download `subaxis` from the MATLAB Exchange, and add the function to the MATLAB path:

```matlab
subaxis_folder = [pwd '\subaxis']; % pwd gets the path to the current folder.
addpath(subaxis_folder);
```

Warning: Name is nonexistent or not a directory: C:\Users\laduran\Desktop\research\presentations\2016_06_07
Figure making\subaxis.
subaxis gives you more flexibility to adjust the appearance of subplots, by defining, for example, the horizontal and vertical distance separating the plots, as well as the size of the right, left, bottom and top margins from the figure:

```matlab
figure('color','w','units','inches','position',[.2 .5 9 9]); hold on;

% Subaxis parameters.
SH = 'SpacingHoriz'; SHv = 0.01; % Horizontal spacing between the subplots.
SV = 'SpacingVert';   SVv = 0.01; % Vertical spacing between the subplots.
MR = 'MarginRight';  MRv = 0.05; % Size of the margin at the right side of the figure.
ML = 'MarginLeft';   MLv = 0.05; % Size of the margin at the left side of the figure.
MT = 'MarginTop';    MTv = 0.05; % Size of the margin at the Top of the figure.
MB = 'MarginBottom'; MBv = 0.05; % Size of the margin at the Bottom of the figure.

cmapNum  = 100; % Numbers of colormaps to create.
colorNum = 10;  % Number of colors per colormap.

for jj = 1:cmapNum
    % Define vectors where we simultaneously change Hue, Saturation and Brightness.
    Brightness = 0.95:-(0.95-0.5)/cmapNum:0.5; % From 0.5 to 0.95.
    Saturation = 0.1:(0.9-0.1)/colorNum:.9; % From 0.1 to 0.9.
    Hue = 0:1/colorNum:1; % From 0.1 to 1.0.

    subaxis(10,10,jj,SH,SHv,SV,SVv,MR,MRv,ML,MLv,MT,MTv,MB,MBv); hold on;
    for ii = 1:colorNum
        % Define a color in the hsv scale and convert to rgb.
        rgb_color = hsv2rgb([mod(Hue(ii) + jj/cmapNum,1) Saturation(ii) Brightness(ii)]);
        % Hue Saturation Value/Brightness.

        % Create ColorNum random datasets, with 20 elements each.
        thetal = 2*pi*rand(20,1); % Generate 20 random angles between 0 and 2*pi.
        radius1 = rand(20,1);    % Generate 20 random radii between 0 and 1.
        x1   = radius1.*cos(thetal); % Transform to cartesian coordinates.
        y1   = radius1.*sin(thetal);

        theta2 = 2*pi*rand(1); % Generate 20 random angles between 0 and 2*pi.
        radius2 = 5*rand(1);   % Generate 20 random radii between 0 and 5.
        x2   = radius2.*cos(theta2); % Transform to cartesian coordinates.
        y2   = radius2.*sin(theta2);

        x   = x1+x2; % Generate data inside an unity circle that is inside a circle
        y   = y1+y2;
        plot(x,y,'o','markerfacecolor',rgb_color,'markeredgecolor','none','markersize',5);
    end

    axis equal;
    xlim([-5.2 5.2]);
    ylim([-5.2 5.2]);
    box on;
end

% Only display Xlabels, Ylabels and ticks in plots located at the bottom and the left of the figure.

fontsize = 9;
if mod(jj,10) == 1 && jj ~= 91
    set(gca,'Xtick',[],'Ytick',[-5 0 5],'YTickLabel',{'-5','0','5'},'FontSize',fontsize);
    ylabel('Y');
elseif jj == 91
    set(gca,'Xtick',[-5 0 5],'XTickLabel',{'-5','0','5'},'FontSize',fontsize);
    ylabel('Y');
    xlabel('X');
elseif jj > 91
    set(gca,'Xtick',[-5 0 5],'XTickLabel',{'-5','0','5'},'FontSize',fontsize);
    xlabel('X');
else
    set(gca,'Xtick',[],'Ytick',[],'FontSize',fontsize);

end
```
by Leonardo A. Sepulveda. Last version: 06/15/2015