MATLAB

Class 4: plotting

Plotting Tips

The first thing you need to consider when plotting is your **message**, the **audience**, and type of **plot**

Making a graphic is always helpful for a reader but not always necessary. Can you say this image in words?

Plotting Tips

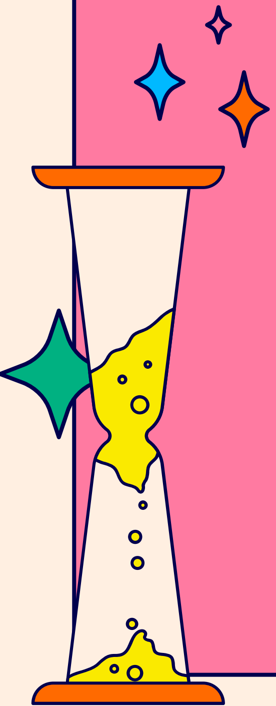
In a general sense the type of plot you pick is very important

Bar plots and line graphs are simple and effective ways to convey a message and are readable by almost all audiences … but they are not always the best option

Plotting Tips

It is important to remember there is a wide range of scientific literacy in the world

Your audience may not be accustomed to reading plots, specifically complex figures



Plotting Tips

- Try plotting information with the **least** amount of **ink** as possible
- Do not overcrowd graphs, give each one room to breath
- Colour choice is important and can make figures misleading

Plotting Tips

Do not mislead readers, be careful about adjusting axes to exaggerate effects

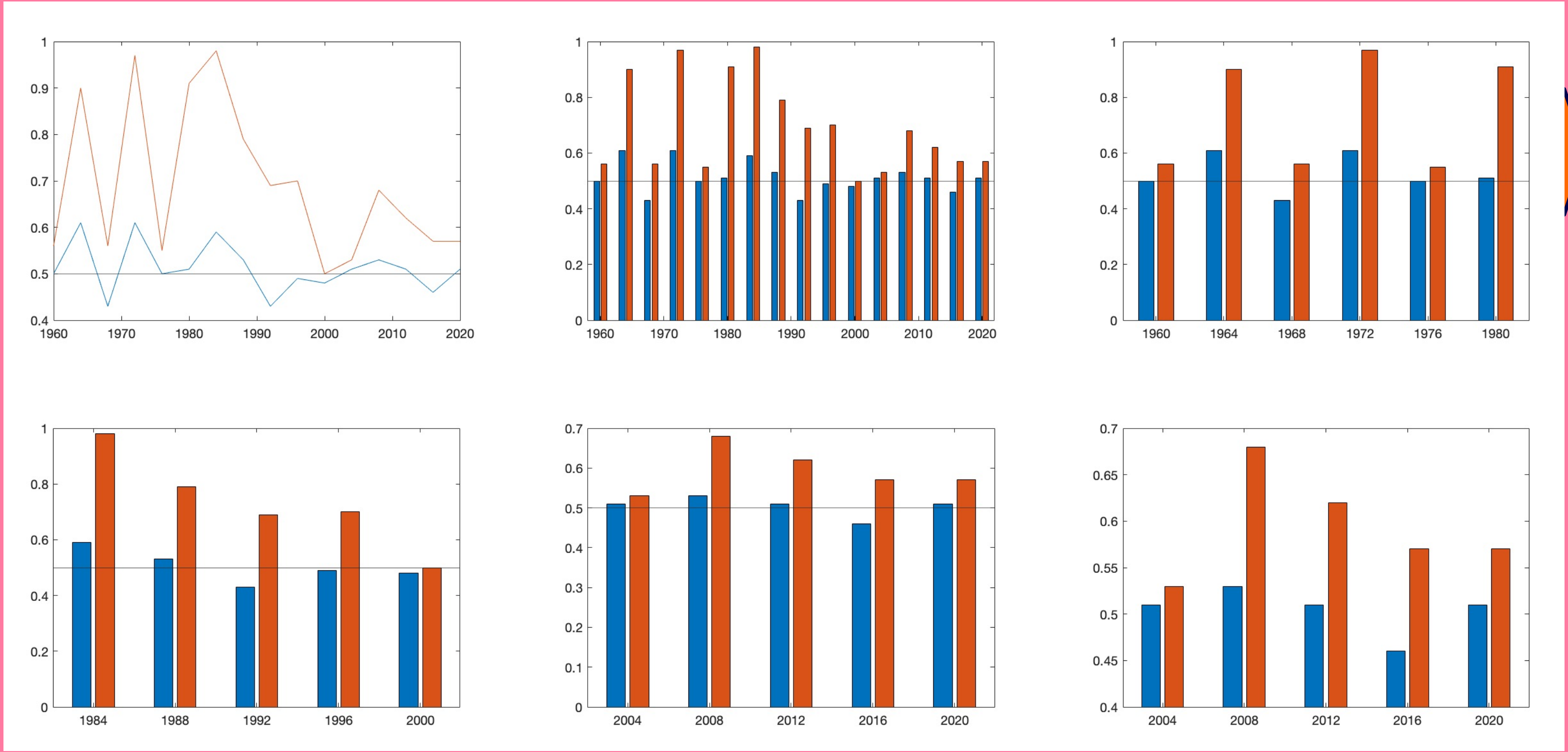
Label everything clearly including axes and units

Always plot **confidence intervals** or a measure of data spread / uncertainty

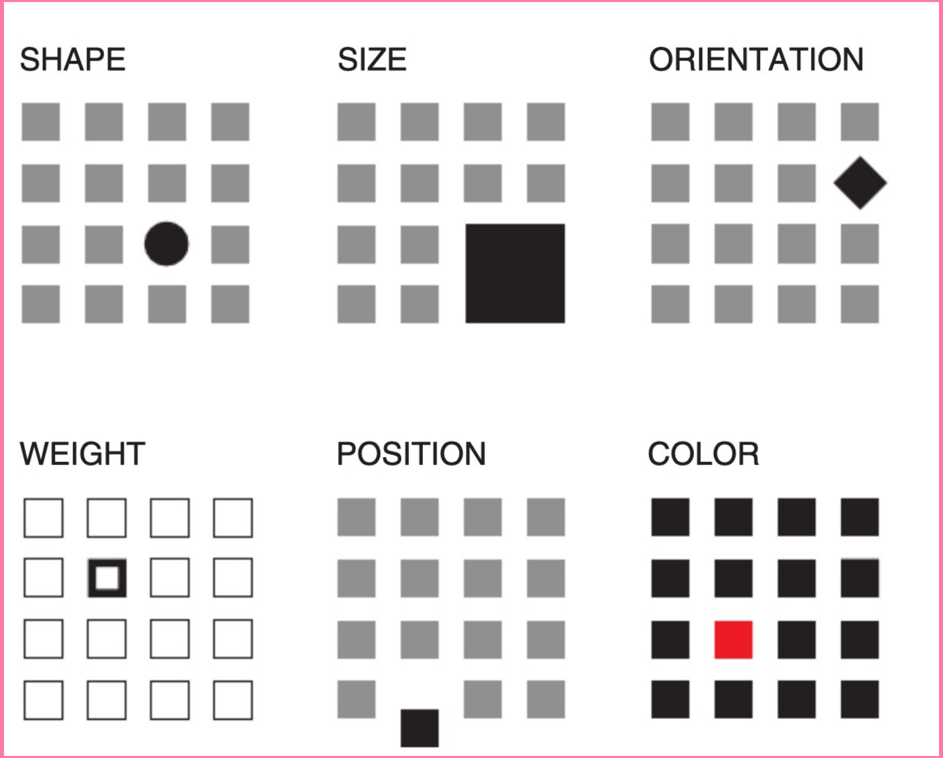
Plotting Tips

Your plot should highlight your data in the best way possible: do not plot a histogram for a binary outcome variable or a bar plot if your data is continuous

Plots tell a story



Plotting Tips



Rolandi et al., 2011

Plot function

The **plot()** function in MATLAB takes in x, y values and returns a line plot. Each element of the plot can be manipulated using different specifiers

Plot function

After running **plot()** you can manually alter aspects of the resulting figure

Xlabel(‘time’)

Ylabel(‘money’)

Title(‘Time is money’)

Plot function

If you have provided multiple sets of information to the plot function or a matrix, you can use the **legend()** function to label each element you've plotted

Plot() assumes the rows are observations and the columns are the different conditions for the legend

Plot Colours

You can change the colour of your lines by specifying one of the following colours from the table

e.g., plot(x, y, 'r')

y	yellow
m	magenta
c	cyan
r	red
g	green
b	blue
w	white
k	black

you can also specify RGB values using a specifier, see below

Plot Markers

You can change the markers of your data by specifying one of these

e.g., plot(x, y, 'x')

'o'	Circle
'+'	Plus sign
'*'	Asterisk
'.'	Point
'x'	Cross
'_'	Horizontal line
'|'	Vertical line
's'	Square
'd'	Diamond
'^'	Upward-pointing triangle
'v'	Downward-pointing triangle
'>'	Right-pointing triangle
'<'	Left-pointing triangle
'p'	Pentagram
'h'	Hexagram

Plot Lines

You can change the appearance of the lines of a plot()

by specifying one of these

e.g., plot(x, y, '-.') these can be combined with markers

and colours

Plot(x, y, 'x-.r')

-	Solid line
--	Dashed line
:	Dotted line
-.	Dash-dot line

Plot other specifiers

‘MarkerSize’, size

‘LineWidth’, size

‘MarkerEdgeColor’, [R G B alpha]

‘MarkerFaceColor’, [R G B alpha]

‘Color’, [R G B alpha]

Etc…

(potential to make a MATLAB plotting cookbook in future)

Figure and close all

I always recommend you begin a new graph by running **figure** this ensures that you are not overwriting any previous information you've plotted before

Reminder that close can be used to **close** currently opened figures

Hold on / off

The command hold on allows one to **add to the existing axes** of a plot you just made. It is like adding another layer.

This does **not** need to be the **same type** of plot

Hold off removes this hold on the figure's axsis and allows you to overwrite them

Other Line plots

If you would like to plot your data in log-log space you can use the function **loglog()** which works essentially the same way that plot does

Useful when data is decaying or exponentially growing

Error Bars

It is always important to give the reader a sense of how uncertain a measure is, whether that be std, std error, or CI

To plot error bars in MATLAB use the function **errorbar()**

Error Bars

errorbar() takes the x, y, and error

All specifiers are like plot() except for 'CapSize'

Use ***'LineStyle'*** to remove line between x values this allows you to plot the error bars **separately** from the underlying graph

Bar graphs

Creates bar plots, see examples in code

Special specifier 'stacked'

Can use **xticks** and **xlabels** to relabel the x-axsis or

change the number of ticks (same for yticks)

Plot but make it fashion

There are many toolboxes in addition to the basic functions of MATLAB, some are developed my MATLAB and others are **external** and need downloading

We will cover some additional methods to plot in MATLAB

Histograms

Allows you to visualize distributions of data

histogram(X, nbins)

‘BarWidth’

‘FaceColor’

‘FaceAlpha’

Scatter

Scatter plot of data inputs take x and y

Same colour and marker specifiers as plot

‘filled’ to colour in marker

Other plots

Loglog, semilogx, semilogy

Boxchart, barh, stairs

Imagesc, polarhistogram

Gramm

I'm not a regular graph, I'm a cool graph

Gramm

Toolbox developed to extend MATLAB's graphing capacities. The code runs much like ggplot in R, whereby data is fed into the gramm function and each layer of the graph is added on top

See below for a cheat sheet summarizing gramm's capacities

https://github.com/piermorel/gramm/raw/master/gramm%20cheat%20sheet.pdf

Gramm

Gramm example script:

```
g=gramm('x',cars.Model_Year,'y',cars.MPG,'color',cars.Cylinders,'subset',cars.Cylinders~=3 & cars.Cylinders~=5);

g.facet_grid([],cars.Origin_Region);

g.geom_point();

g.stat_glm();

g.set_names('column','Origin','x','Year of production','y','Fuel economy (MPG)','color','# Cylinders');

g.set_title('Fuel economy of new cars between 1970 and 1982');

Figure('Position',[100 100 800 400]);

g.draw();
```

See example on their website

References

- **Rolandi** et al 2011. A Brief Guide to Designing Effective Figures for the Scientific Paper. *Advanced Materials*
- **Rougier** et al 2014. Ten Simple Rules for Better Figures. *Plos Computational Biology*
- ***Nature** blog http://blogs.nature.com/methagora/2013/07/data-visualization-points-of-view.html*