Improving MATLAB plots

with information relevant to plotting in all programs!

Clear, professional plots

- A good plot can often communicate the intended message on its own (even without caption, or corresponding text in body)

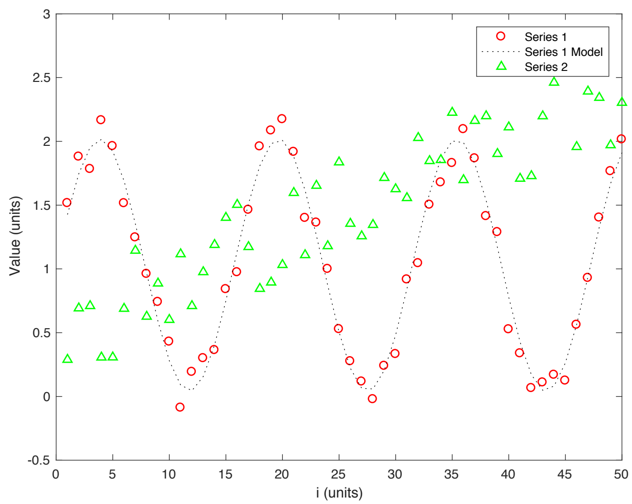
- Your reader determines whether your work is clear, not you.

- Before you plot, ask who is your reader? What is the medium?
  - Presentation (e.g. Powerpoint, Keynote, Beamer, etc.)
  - Paper publication (e.g. p-set, report, thesis, journal, etc.)
  - Poster session
  - Email only (e.g. quick update for professor, sponsors, collaborators, etc.)
  - Color? Black and white? Animated? Static?

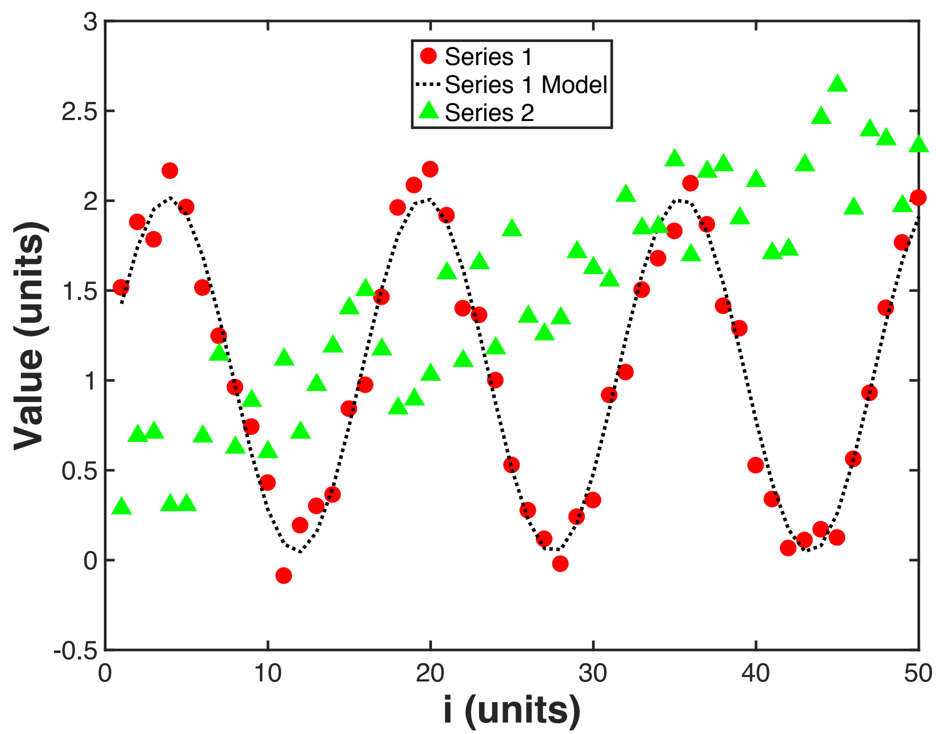
Best practices

- White background
- Box enclosing plot area
- Data points for sparse data, lines for dense or high freq. data
- Thick lines, large data point markers
- Try to incorporate visual redundancy: each data series has at least two different unique properties (e.g. marker shape, colour, etc.)
- Assume your plot might be printed at some point
  - Use thick lines and colour schemes that map well to grayscale printing
- Include units with axes labels
- Don’t use: 3D pie charts, 3D bar charts
- Avoid pie charts: stacked bar charts are better to compare proportions

Best practices in practice



MATLAB default



MATLAB default with improvePlot()

MATLAB advice

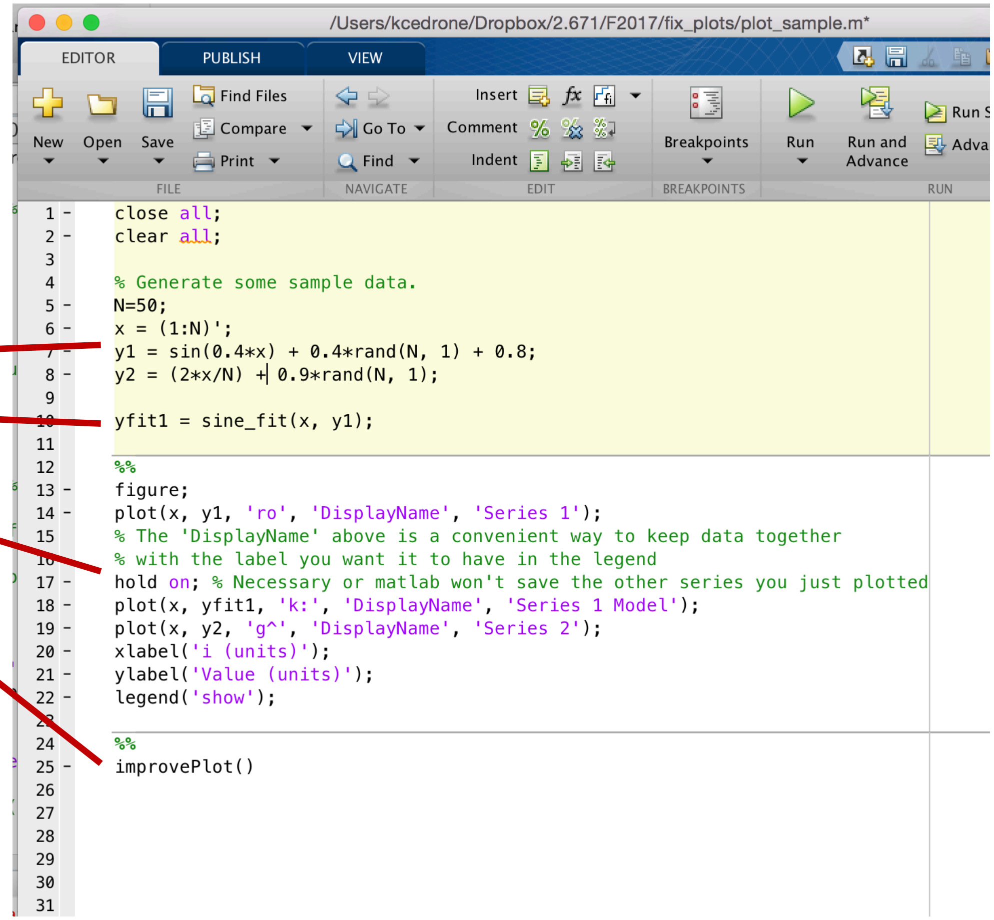
- Use MATLAB to generate plots that are 80-90% ready for publication. Then, you will manually complete the last 10-20%
  - You may be tempted to have MATLAB make every plot perfect automatically
  - This will be frustrating when MATLAB eventually makes a mistake with an edge/corner case
  - Journals or poster sessions often have unique requirements

- Don’t rush! If you don’t have time to do it correctly, when will you have time to do it over again?

- Get a repeatable system in place so you can reuse your tools

MATLAB process

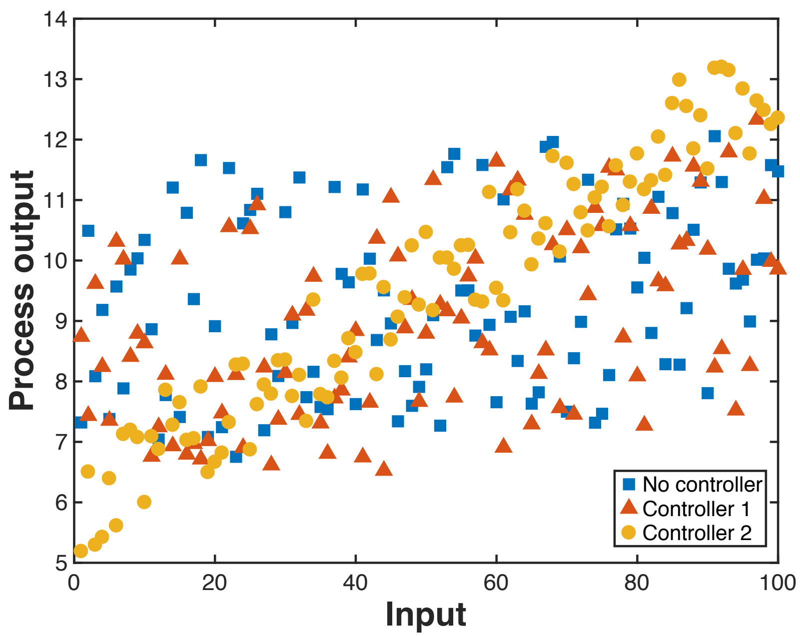
1. Rough plot
   1. Import data from model or experiment
   2. Prepare data (e.g. filter, unit conversions)
   3. Plot data (label axes, legend, etc.)
2. Major clean-up
   - *improvePlot.m* shortcut!
3. Manual tweaks with MATLAB, Illustrator, Inkscape, etc.
4. Review and revise
5. Export
   1. File >> Save as .fig (can re-open and edit in MATLAB again later)
   2. Edit >> Copy figure (to paste into document)
   3. File >> Save as .png, .bmp **NOT .jpeg**



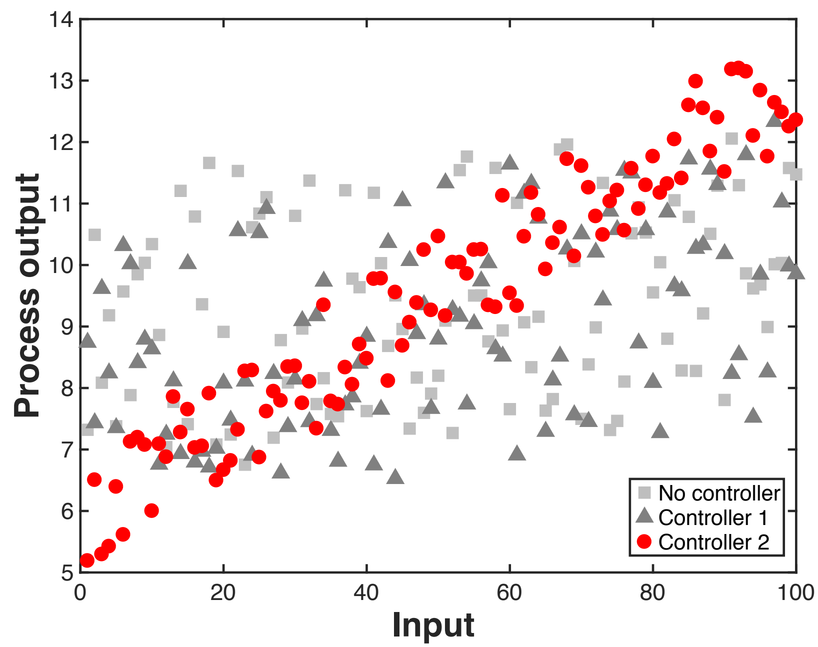
Some specific strategies…

Emphasis: Each plot should tell *part* of the story

- When a plot has **too much** information the message may be unclear.
- Find a way to **add emphasis** (or divide into separate plots)

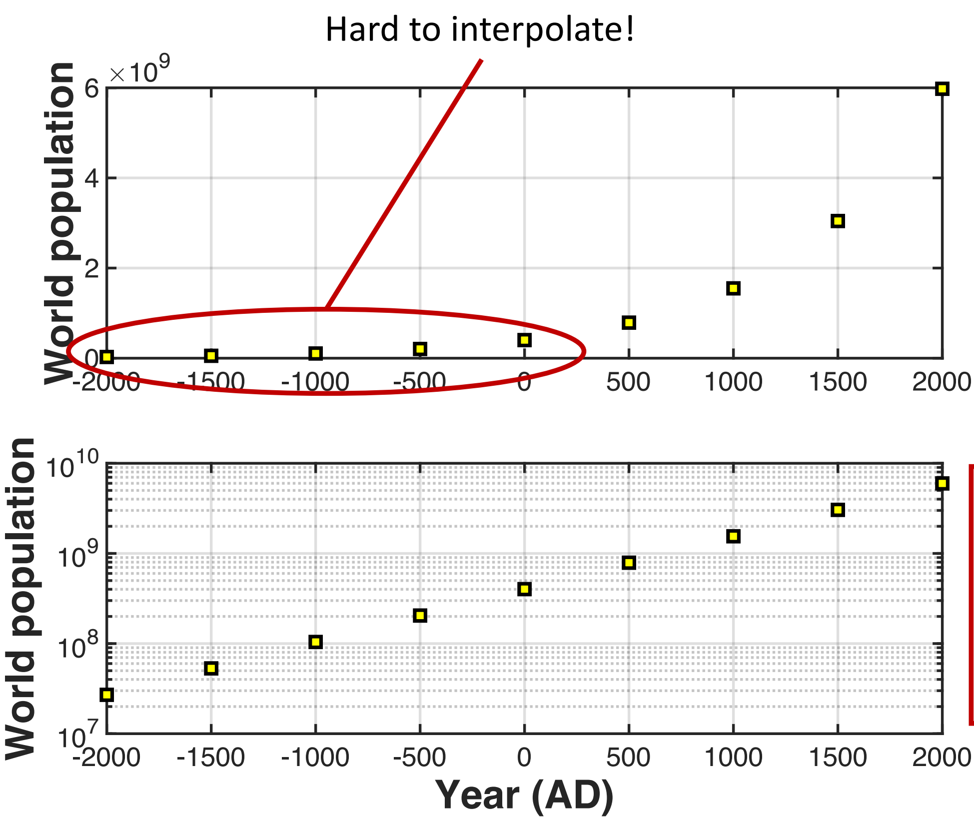


MATLAB default



Color used to emphasize Controller 2

Log (and semilog) plots can include data that span many orders of magnitude



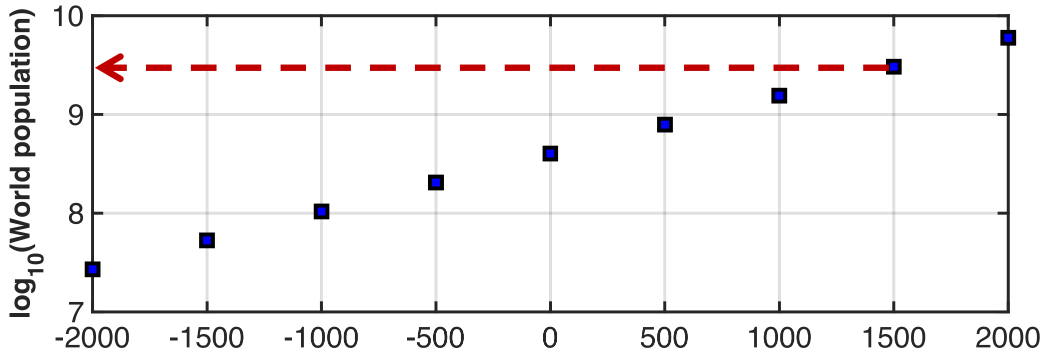

```
%%
t=-2000:500:2020;                % Time, year AD
pop=27e6*exp(0.00135*(t-t(1))); % Crude population model


figure;
subplot(2,1,1);                  % 2 rows, 1 column, plot 1
plot(t, pop, 'ys');
ylabel('World population');
grid on;

subplot(2,1,2);                  % 2 rows, 1 column, plot 2
semilogy(t, pop, 'ys');
ylabel('World population');
xlabel('Year (AD)');             % Label the common x-axis at the bottom
grid on;

improvePlot();
```

semilogy(x,y,…) applies log spacing to y-axis
semilogx(x,y,…) applies log spacing to x-axis
loglog(x,y,…) applies log spacing to both axes

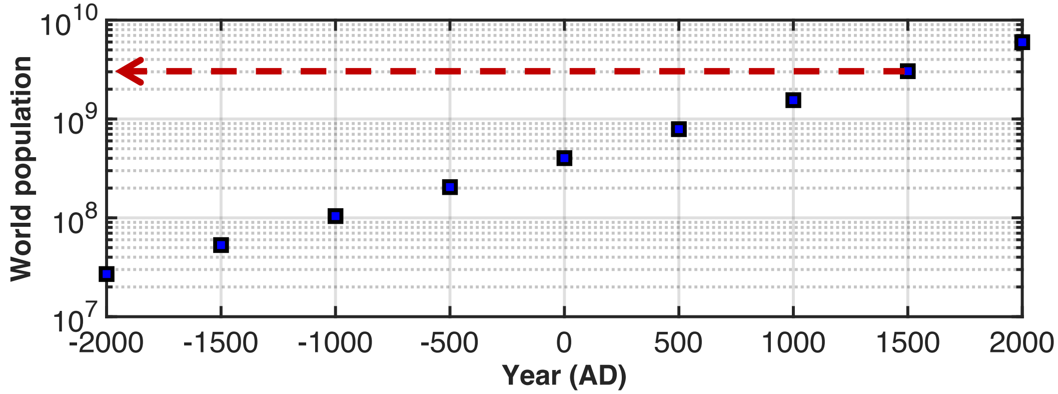
Logarithmic axes, not log(data) on linear axes

A plot of log(data) looks similar, but this data is harder to interpret:

Population in 1500 ~ 10^9.5 = ??



Population in 1500 ~ 3x10^9

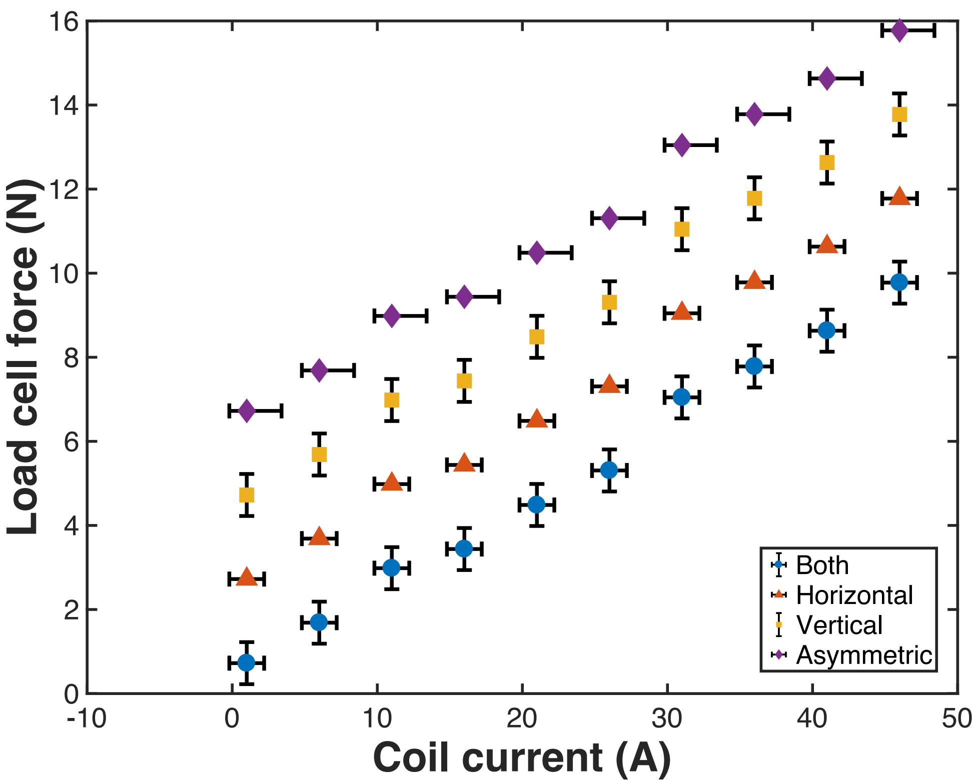


Error bars

In 2.671, we typically use 95% confidence intervals, which are symmetric about the mean

MATLAB can accommodate:

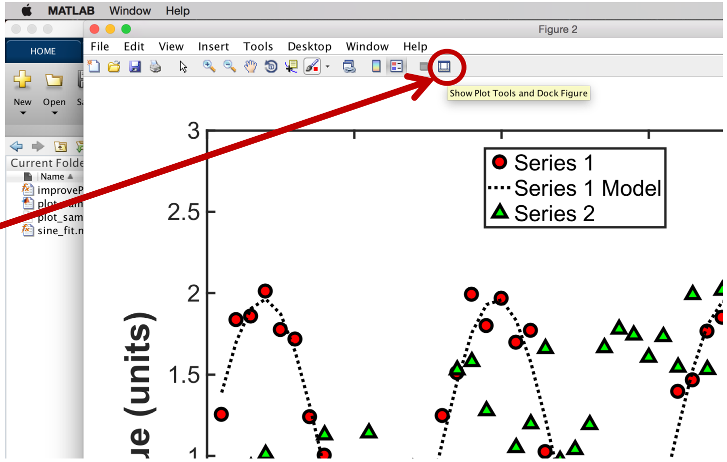
- Vertical, horizontal or vertical and horizontal error bars
- Symmetric (2.671) and asymmetric error bars



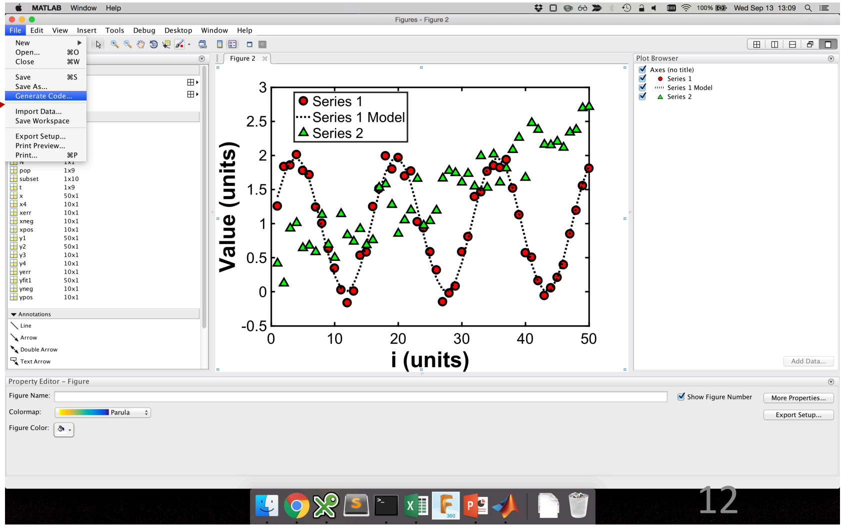

```
%%
figure;
x4 = (1:5:50)';
N = length(x4);
y4 = (2*x4/N) + 0.9*rand(N, 1);
y5 = y4 + 2;
y6 = y5 + 2;
y7 = y6 + 2;

% Here I assume constant error.
% You should substitute a different error model,
% e.g. 95% confidence interval, instrument accuracy, etc.
yerr = 0.5*ones(size(y4));
xerr = 1.2*ones(size(x4));

% Matlab can apply asymmetric or symmetric error bars
yneg=yerr;
ypos=yerr;
xneg=xerr;
xpos=xerr;

errorbar(x4, y4, yneg, ypos, xneg, xpos, 'o', 'DisplayName', 'Both');
hold on;
errorbar(x4, y5, xneg, xpos, '^', 'horizontal', 'DisplayName', 'Horizontal');
errorbar(x4, y6, yneg, ypos, 's', 'vertical', 'DisplayName', 'Vertical');
errorbar(x4, y7, xneg, 2*xpos, 'd', 'horizontal', 'DisplayName', 'Asymmetric');
xlabel('Coil current (A)');
ylabel('Load cell force (N)');
legend('show');

improvePlot();
```

Next steps in MATLAB

- You can modify a plot with MATLAB Plot Tools (e.g. axis limits, titles, line weight, add annotations, etc.)



- File >> Generate Code
- Will recreate the code that would make that exact plot -- you can learn how to make that type of plot yourself, OR, use that function to generate similar plots from new data

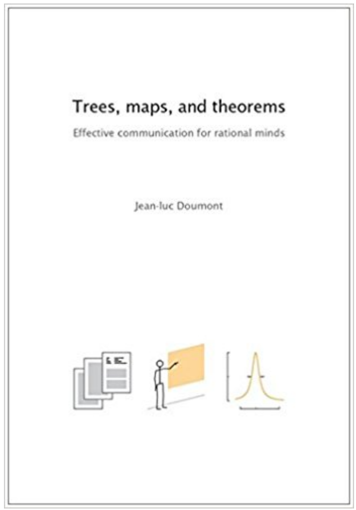


Try it yourself

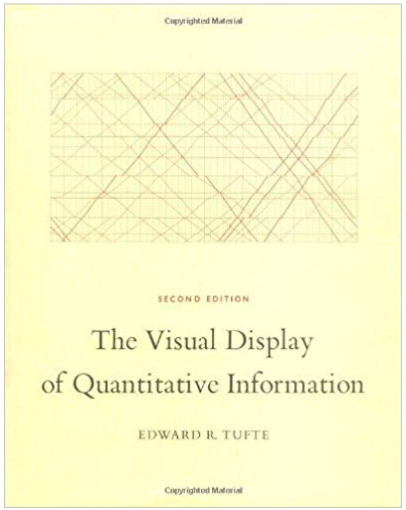
- Download improvePlot.m
- Copy *improvePlot.m* to your MATLAB working directory (or add it to your MATLAB search path)
- Call improvePlot() after you have created a figure
- Modify it to your preferences
- Bugs? Requests? E-mail Dr. Cedrone: kcedrone@mit.edu

Other resources

- 2.671 MATLAB questions – Dr. Kevin Cedrone, kcedrone@mit.edu
- ME CI Instructors



Trees, maps and theorems (2009)
Jean-Luc Dumont



The Visual Display of Quantitative Information (1983)
Edward Tufte