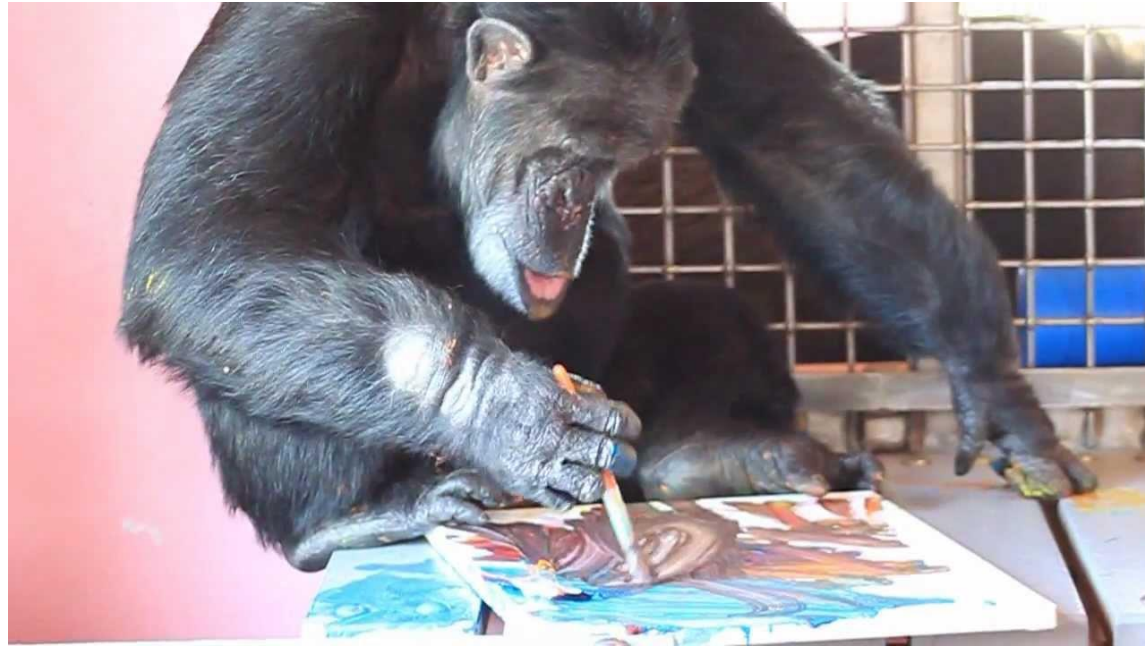


Plot your data

Jeremy Badler, Ph.D.

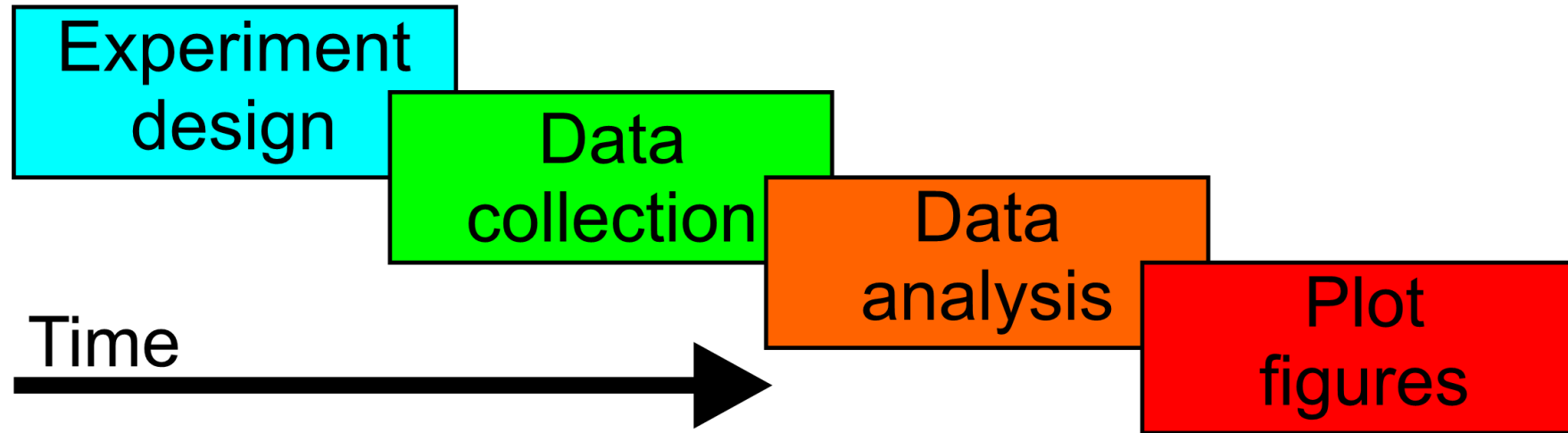
20th January, 2020

Clear presentation is important



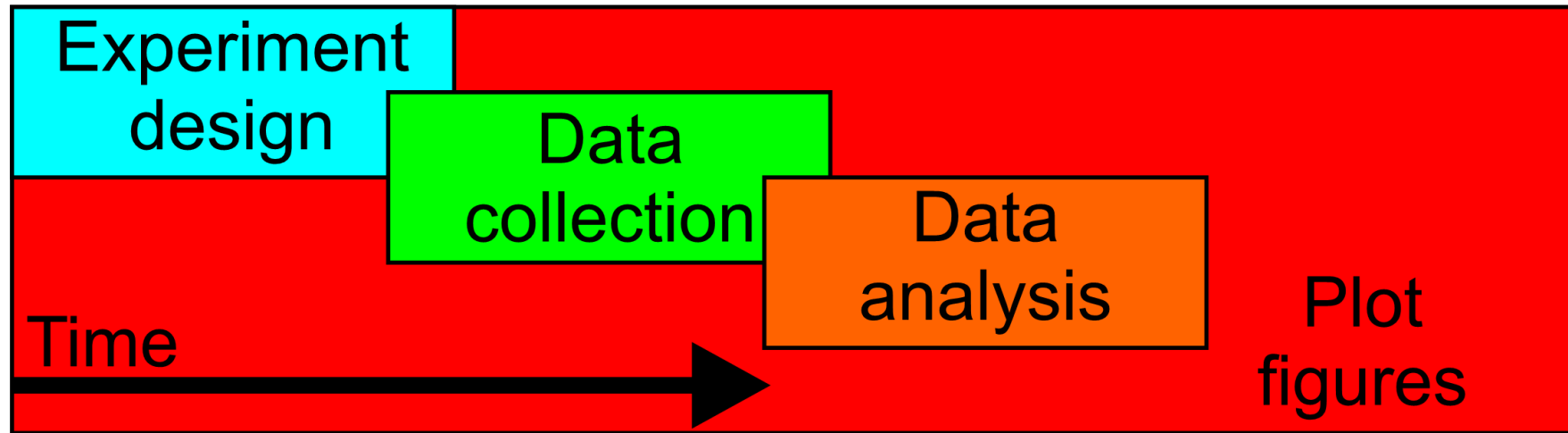
But also clear thinking...

When to plot?



Plotting helps you understand your data

When to plot?



As well as your experiment, analysis, etc.

Outline

- 1) Structure of graphics in MATLAB
- 2) Getting to know your data
- 3) Things to do when plotting your results
- 4) Things to avoid when plotting results
- 5) Sharing your data with the world

0) How to make a Methods figure

- is unfortunately beyond the scope of this presentation

citation unknown, if anyone can find I will be forever grateful

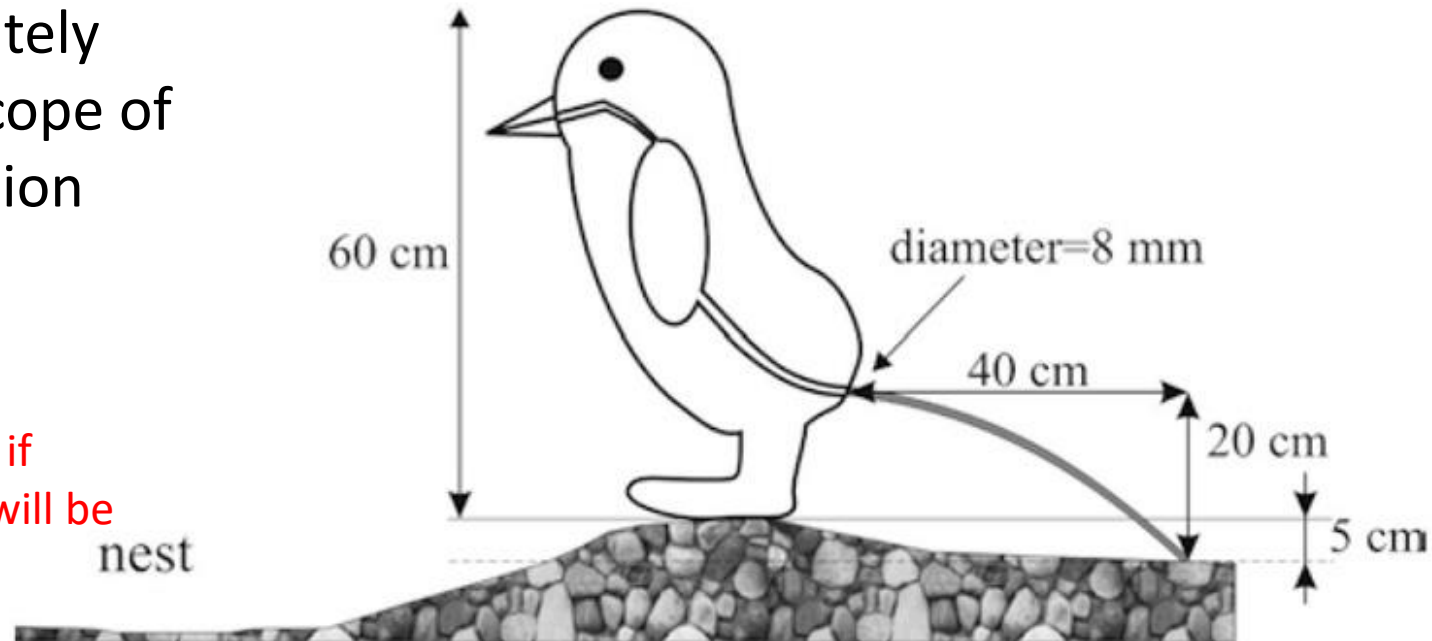


Fig. 1 Position of model penguin during defaecation and physical parameters used to calculate rectal pressure necessary to expel faecal material over a distance of 40 cm

1) Structure of graphics in MATLAB

Here `h` is the handle of the line object, while `h1` is the handle of the axes object. Now, if you type the following `get` commands, you obtain a list of the line properties and a list of the axes properties:

Example 1	Example 2
<pre>>> get(h) Color: [1 0 0] EraseMode: 'normal' LineStyle: '-' LineWidth: 0.5000 Marker: '.' MarkerSize: 6 MarkerEdgeColor: 'auto' MarkerFaceColor: 'none' XData: [1x63 double] YData: [1x63 double] ZData: [1x0 double] BeingDeleted: 'off' ButtonDownFcn: [] Children: [0x1 double] Clipping: 'on' CreateFcn: [] DeleteFcn: [] BusyAction: 'queue' HandleVisibility: 'on' HitTest: 'on' Interruptible: 'on' Selected: 'off' SelectionHighlight: 'on' Tag: '' Type: 'line' UIContextMenu: [] UserData: [] Visible: 'on' Parent: 158.0052 DisplayName: '' XDataMode: 'manual' XDataSource: '' YDataSource: '' ZDataSource: ''</pre>	<pre>>> get(h1) BackgroundColor = none Color = [0 0 0] EdgeColor = none EraseMode = normal Editing = off Extent = [3.12 -1.21 0.69 0.09] FontAngle = normal FontSize = [10] FontUnits = points FontWeight = normal HorizontalAlignment = center LineStyle = - LineWidth = [0.5] Margin = [2] Position = [3.48 -1.13 1.00] Rotation = [0] String = time [s] Units = data Interpreter = tex VerticalAlignment = cap BeingDeleted = off ButtonDownFcn = Children = [] Clipping = off CreateFcn = DeleteFcn = BusyAction = queue HandleVisibility = off HitTest = on Interruptible = on Parent = [158.005] Selected = off SelectionHighlight = on Tag = Type = text UIContextMenu = [] UserData = [] Visible = on</pre>

- Hierarchical
 - Workspace (root)
 - Figures
 - Axes
 - Objects

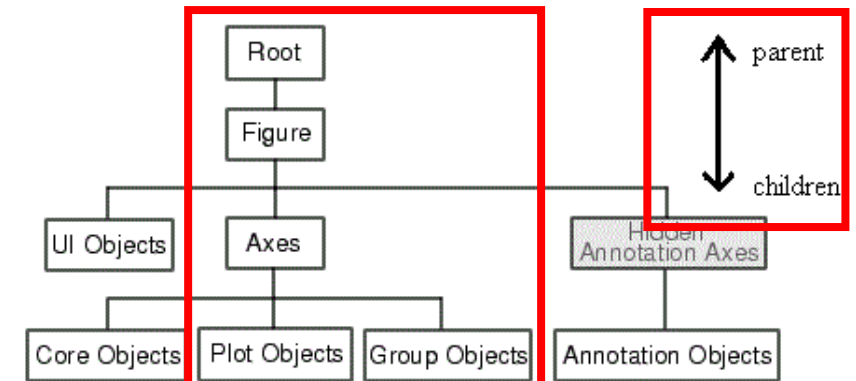
- Lists of properties

```
get(h, 'Marker');
```

```
set(h, 'Marker', 'o');
```

```
h.Marker = 'o';
```

```
line([0,0], ylim, 'Color', 'k');
```



Some properties are more useful than others

Here `h` is the handle of the line object, while `h1` is the handle of the axes object.
Now, if you type the following `get` commands, you obtain a list of the line properties
and a list of the axes properties:

Example 1	Example 2
<pre>>> get (h) Color: [1 0 0] EraseMode: 'normal' LineStyle: '-' LineWidth: 0.5000 Marker: '.' MarkerSize: 6 MarkerEdgeColor: 'auto' MarkerFaceColor: 'none' XData: [1x63 double] YData: [1x63 double] ZData: [1x0 double] BeingDeleted: 'off' ButtonDownFcn: [] Children: [0x1 double] Clipping: 'on' CreateFcn: [] DeleteFcn: [] BusyAction: 'queue' HandleVisibility: 'on' HitTest: 'on' Interruptible: 'on' Selected: 'off' SelectionHighlight: 'on' Tag: '' Type: 'line' UIContextMenu: [] UserData: [] Visible: 'on' Parent: 158.0052 DisplayName: '' XDataMode: 'manual' XDataSource: '' YDataSource: '' ZDataSource: ''</pre>	<pre>>> get (h1) BackgroundColor = none Color = [0 0 0] EdgeColor = none EraseMode = normal Editing = off Extent = [3.12 -1.21 0.69 0.09] FontAngle = normal FontName = Helvetica FontSize = [10] FontUnits = points FontWeight = normal HorizontalAlignment = center LineStyle = - LineWidth = [0.5] Margin = [2] Position = [3.48 -1.13 1.00] Rotation = [0] String = time [s] Units = data Interpreter = tex VerticalAlignment = cap BeingDeleted = off ButtonDownFcn = Children = [] Clipping = off CreateFcn = DeleteFcn = BusyAction = queue HandleVisibility = off HitTest = on Interruptible = on Parent = [158.005] Selected = off SelectionHighlight = on Tag = Type = text UIContextMenu = [] UserData = [] Visible = on</pre>

- Color
- FaceColor, EdgeColor
- LineStyle
- LineWidth
- Marker
- MarkerSize
- MarkerFaceColor
- XData, YData

- FontAngle, FontName, FontSize
- Interpreter ('tex', 'none')
- XTick, YTick
- *TickLabel, *TickLabel Rotation
- *Scale ('linear', 'log')
- *Dir ('normal', 'reverse')
- FaceAlpha (transparency)

Some examples of using properties

```
h.MarkerFaceColor = h.MarkerEdgeColor; % filled points
```

```
title('Block_A', 'Interpreter', 'none'); % display underscore correctly
```

```
set(gca, 'XTick', [0 1], 'XTickLabel', {'Left', 'Right'}, ...
```

```
    'XTickLabelRotation', 90); % axis categorical labels
```

```
set(gca, 'YScale', 'log'); % post-hoc semilogy plot
```

```
plot(xMeshGridPix, yMeshGridPix, '.', 'YDir', 'reverse');
```

```
    % plot with y counting down like PTB
```

```
h.LineStyle = 'none'; % remove lines, e.g. for error bars
```



2) Getting to know your data

- Make quick plots of results (& parameters)
 - Are the subjects doing the task correctly?
 - Is my randomization / algorithm working correctly?



- Quick histogram to check distributions
`hist(dataVector, nBins);` % nBins is optional, but default only = 20
- Quick cumulative frequency plot
`plot(sort(dataVector), 1:length(dataVector), 'Marker', '.');` % line + points ('.-')
- Quick overlay plots for mismatched data
`plot(dataVectorSubj1); hold on; plot(dataVectorSubj2);`
- Plot each subject individually before population summaries

3) Things to do when plotting your results



- Plotting errorbars

`errorbar(xVector, yVector, yNegExtent, yPosExtent);` **OR**

`line(matrixX, matrixY)` where `matrix*` = [firstPointStart, secondPointStart, ...;
firstPointEnd, secondPointEnd, ...];

- Plotting histograms

`[binCounts, binEdges] = histcounts(data, <binning options>);`

`binCenters = binEdges(1:end-1) + diff(binEdges); bar(binCenters, binCounts);`

- Cumulative plots

`yPercentile = linspace(0, 100, length(data)); plot(sort(data), yPercentile);`

- Controlling axes

`xlim([0 100]); set(gca, 'YLim', [0, 100]); axis equal`

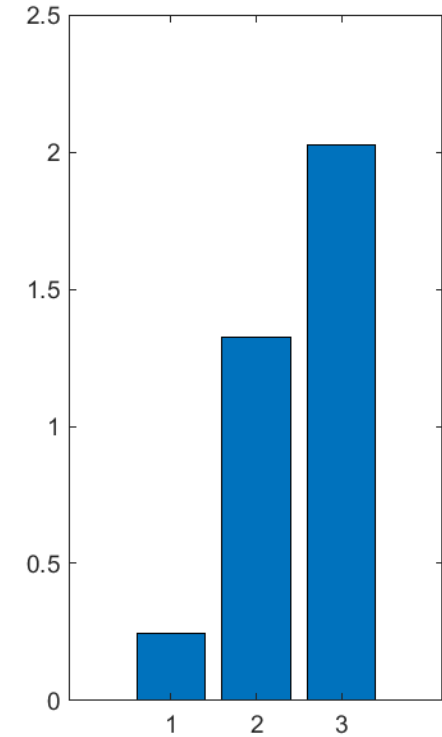
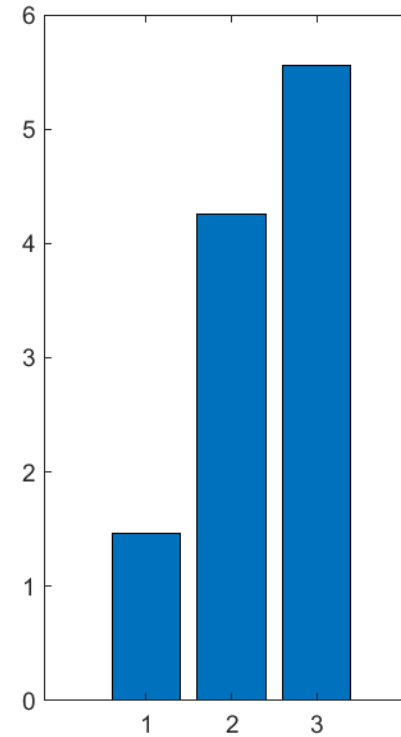
Additional considerations for temporal data



- Look at **every** trial at least once to get a feel for performance, reliability, noise level, etc.
- For each subject, plot all trials of <metric of interest> vs time.
 - Color code by condition
 - Consider overlaying average traces (plot **last**)
 - Consider overlaying target trace(s), if applicable
 - Consider event markers (did you remember to align?)
- Collapse multidimensional or nonuniform metrics to uniform linear scale
 - e.g., map “xy position” (2-D, nonuniform) into “distance from goal”
- Consider ways to collapse temporal data for comparing across subjects
 - e.g., average metric value over specific time interval

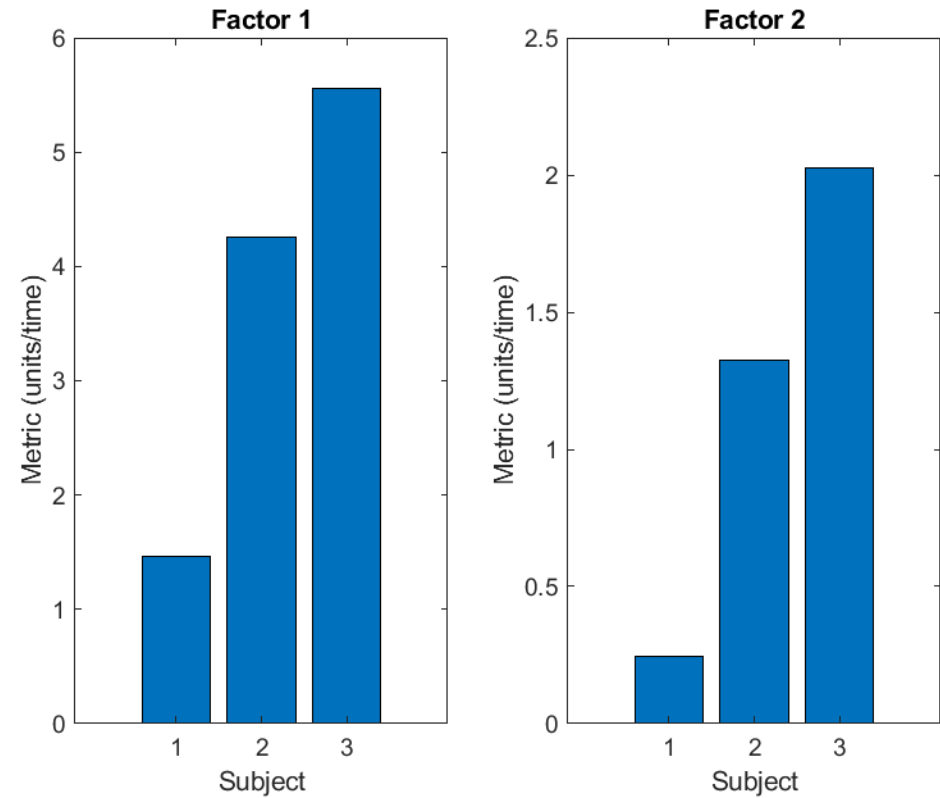
4) Things to avoid when plotting results

- Missing axis labels, titles etc.



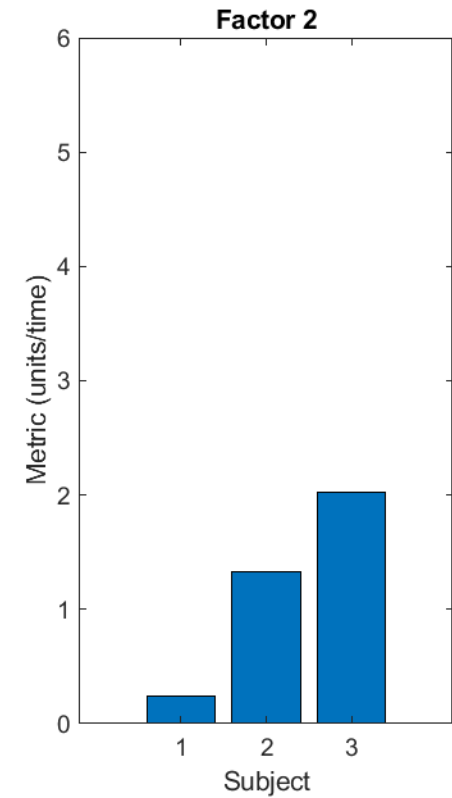
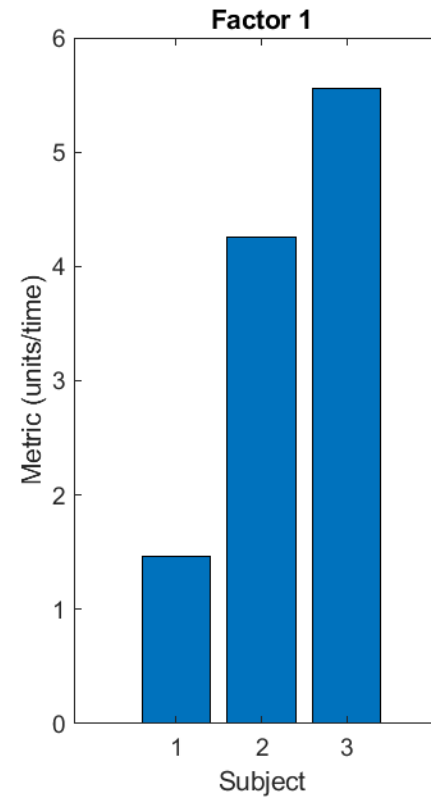
4) Things to avoid when plotting results

- Missing axis labels, titles etc.
- Unequal axes for comparative data



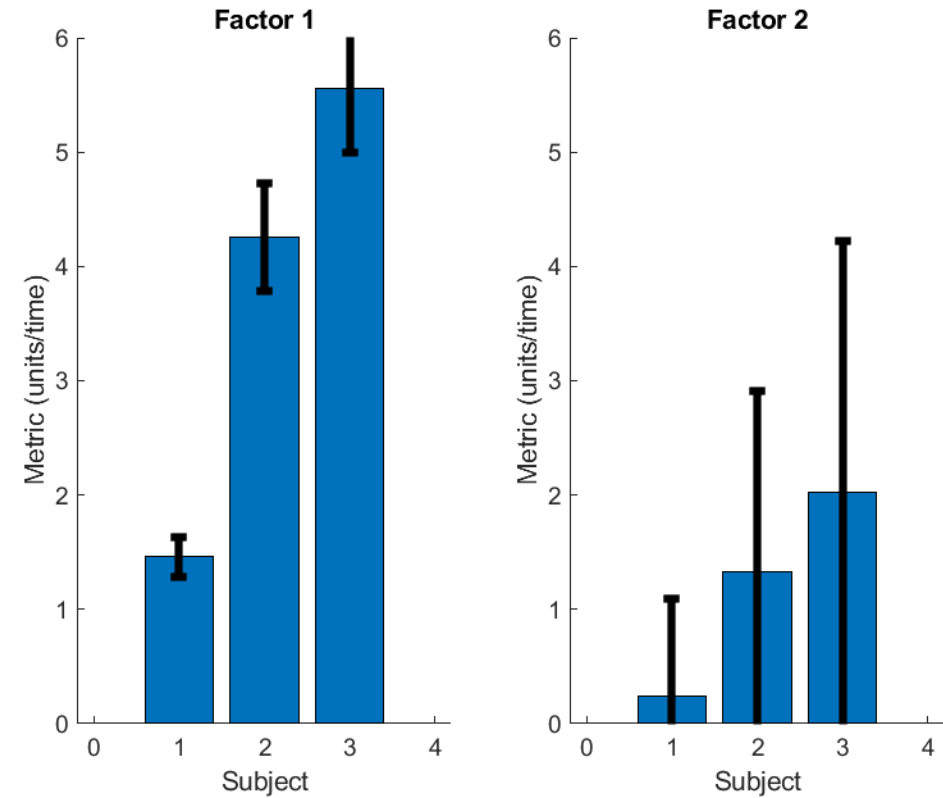
4) Things to avoid when plotting results

- Missing axis labels, titles etc.
- Unequal axes for comparative data
- **Missing error bars**



4) Things to avoid when plotting results

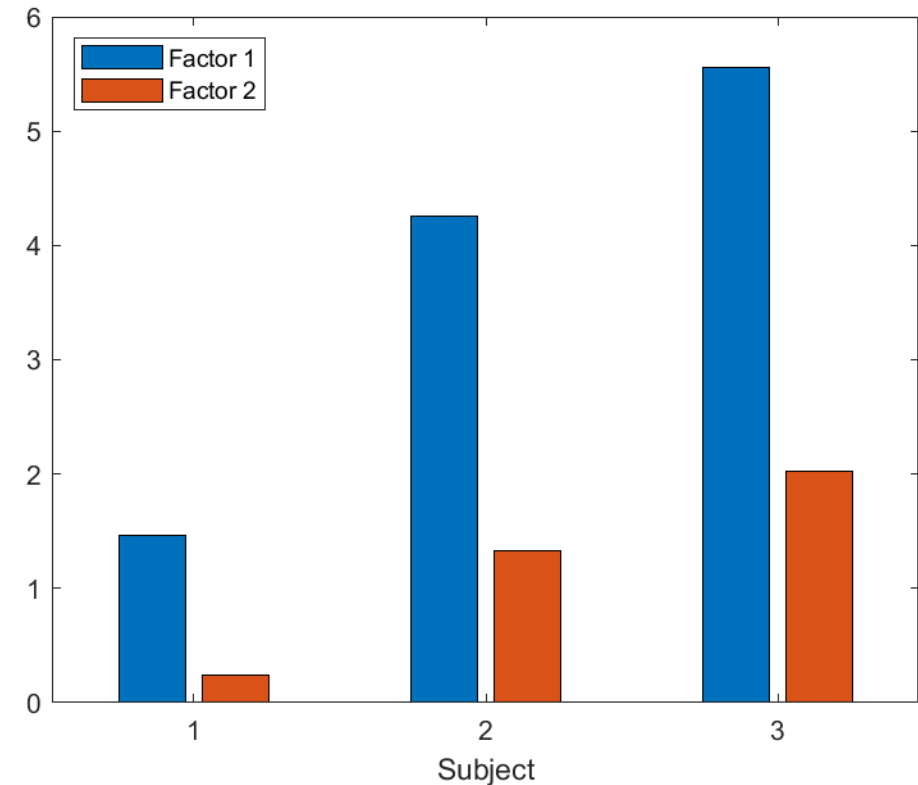
- Missing axis labels, titles etc.
- Unequal axes for comparative data
- Missing error bars
- Emphasize wrong comparisons



compare subjects within factors

4) Things to avoid when plotting results

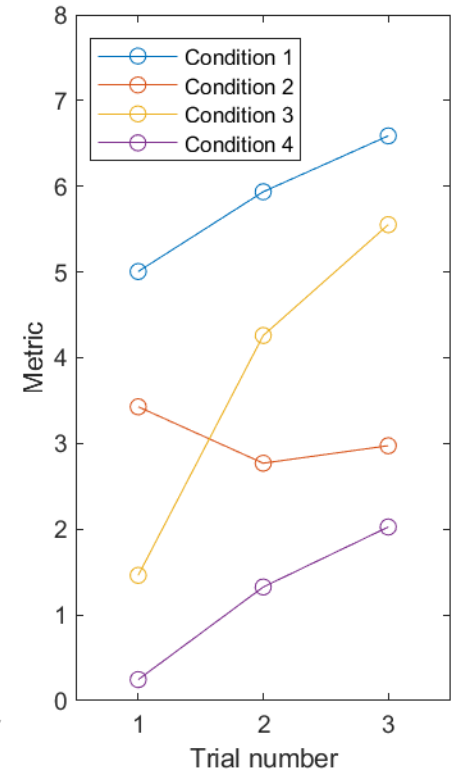
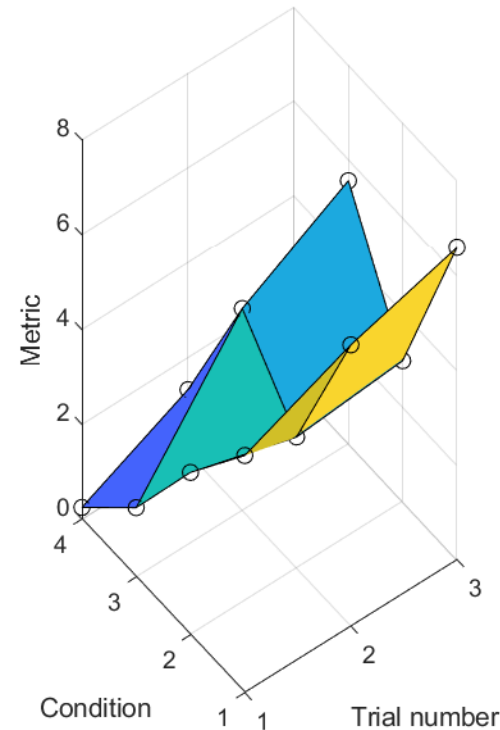
- Missing axis labels, titles etc.
- Unequal axes for comparative data
- Missing error bars
- **Emphasize wrong comparisons**



compare factors within subjects

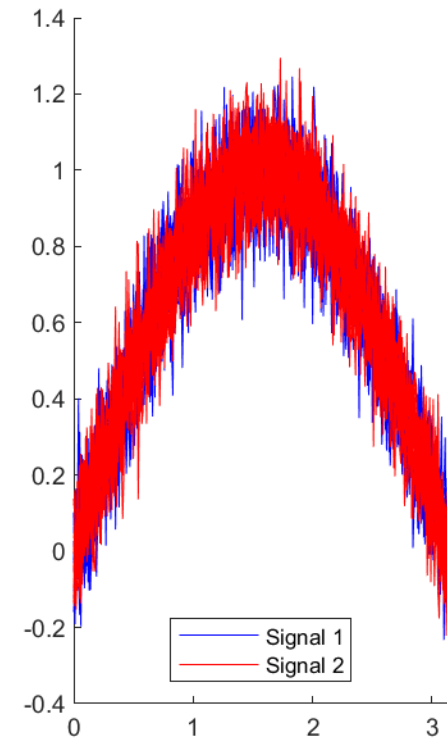
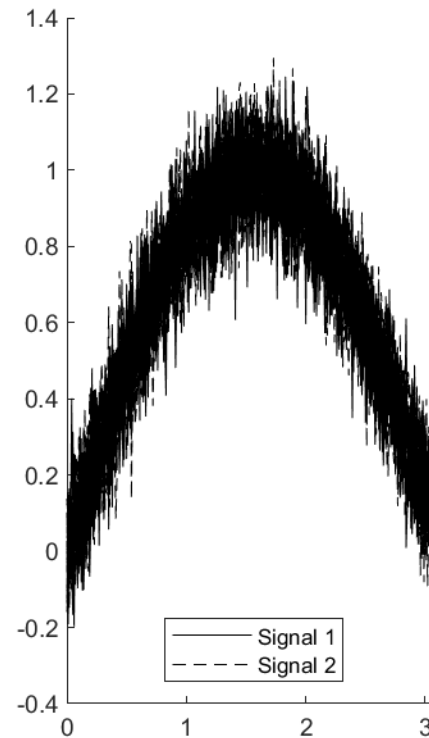
4) Things to avoid when plotting results

- Missing axis labels, titles etc.
- Unequal axes for comparative data
- Missing error bars
- Emphasize wrong comparisons
- **Use 3-D when 2-D will do**



4) Things to avoid when plotting results

- Missing axis labels, titles etc.
- Unequal axes for comparative data
- Missing error bars
- Emphasize wrong comparisons
- Use 3-D when 2-D will do
- Inconsistent or insufficient colors
- Data obscured by draw order



5) Sharing your data with the world

- Saving your graphics

- Save as MATLAB figure file (.fig) – not portable or editable. Not recommended.
- Save as bitmap (.png) – portable but not editable. Good for quick data sharing!

- `save -dpng PrelimRespFreqPlotSubj01`

- For multiple plots, just arrange them on the screen and do a screenshot!

- This is one of the few things Windows Paint is good for (crop & save).

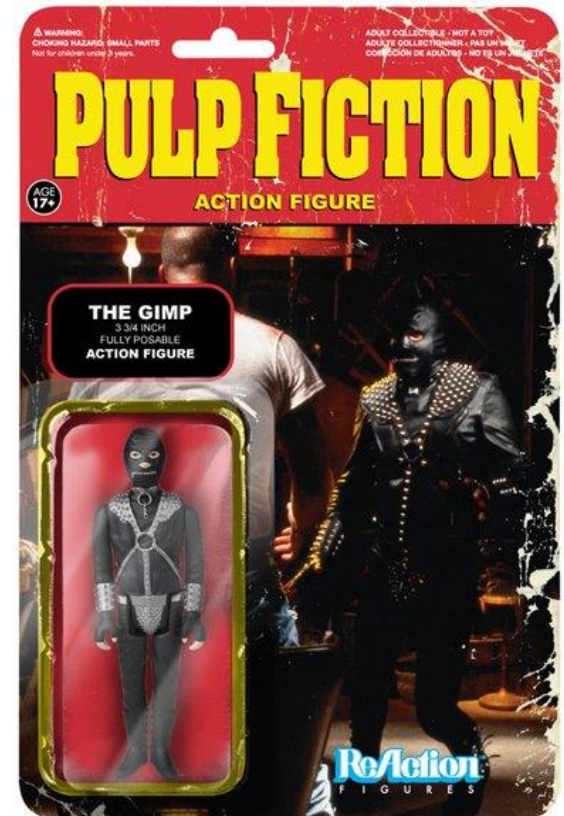
- Save as vector format (.ps) – not easily viewable, but editable. Good for final formatting (see next slide)

- `save -dpsc2 RespFreqPlotAllSubjs`



Post-processing

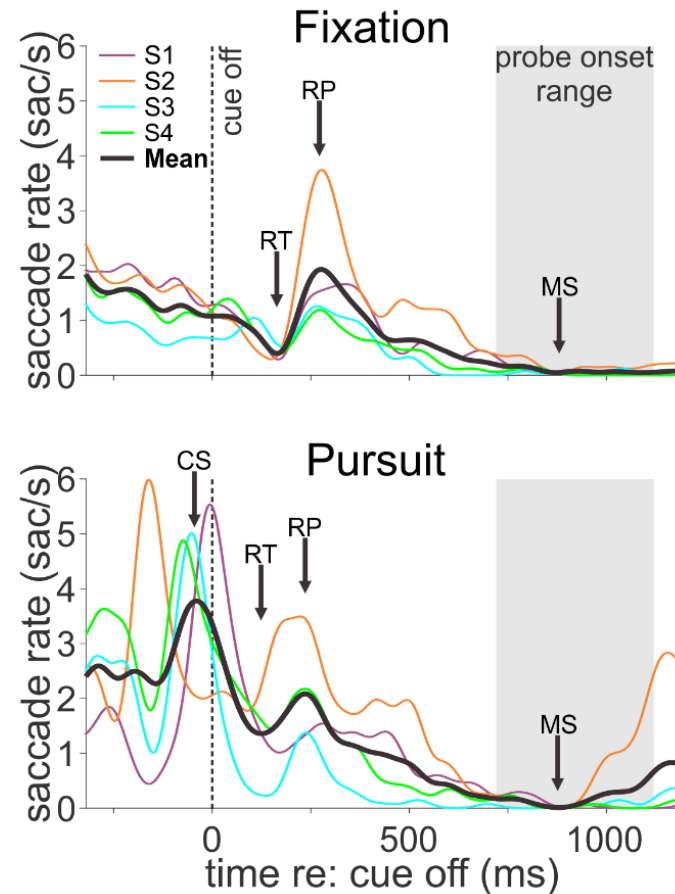
- MATLAB can do a lot, but you really need a CAD program
 - CorelDraw, Adobe Illustrator, GIMP
- Complex layouts
 - Insets, overlays, marginal histograms, posters...
- Spot fixes without rerunning code
 - Labels, colors, spacing, single-dimension resize....
- You need something for the rest of your figures
 - Methods, models, flow diagrams...
- Seriously, don't be one of those PowerPoint people



Examples of post-processed figures

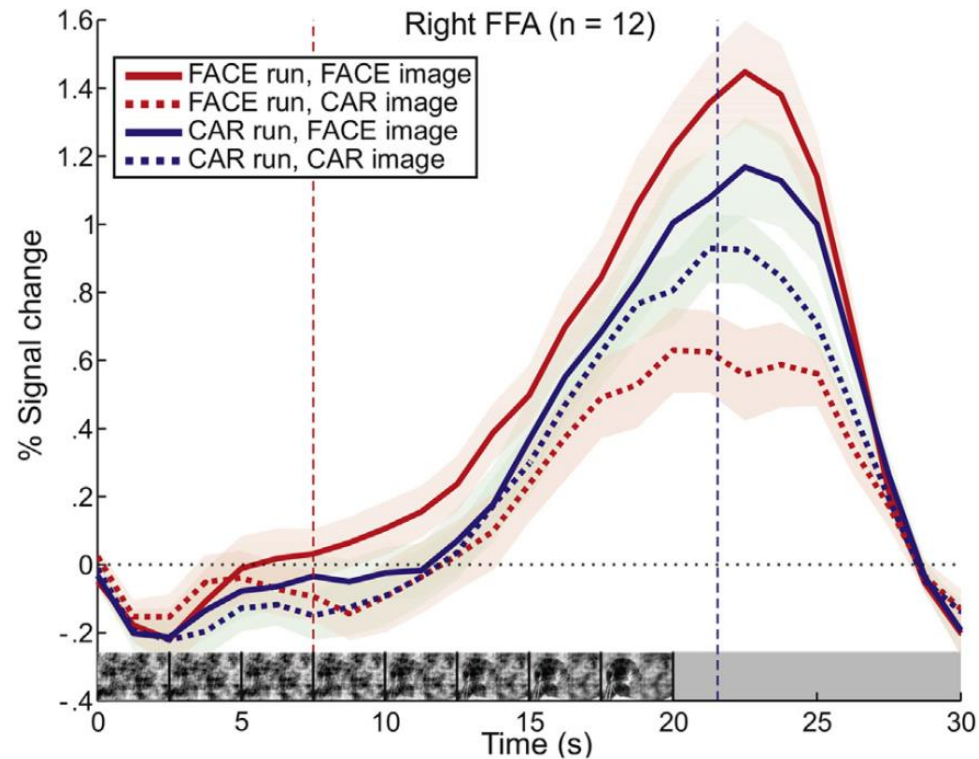
Examples of post-processed figures

- Multiple panels, average traces, event markers



Examples of post-processed figures

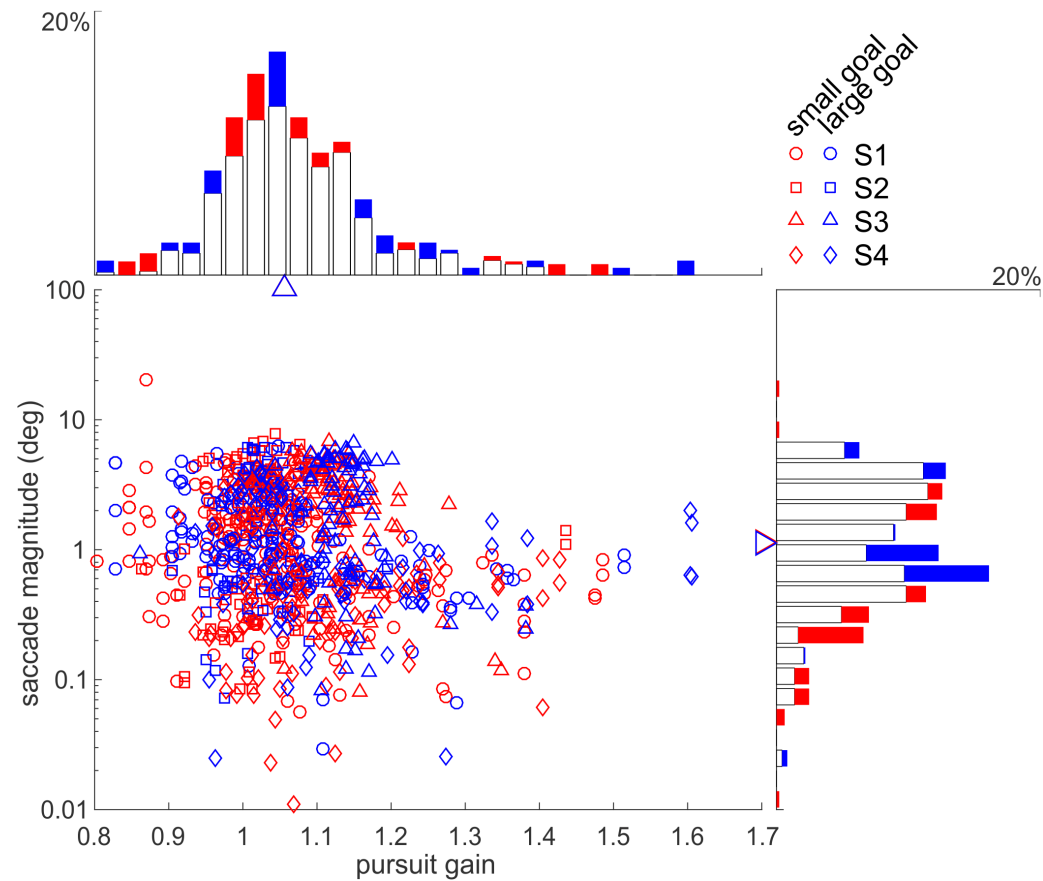
- Shaded overlay with methods inset



Jiang, Badler, Righi & Rossion 2015

Examples of post-processed figures

- Multipart figure with composite histograms & customized legend



Examples of post-processed figures

- Multiple parts, event markers, methods inset, offset axis scale bars, two temporal alignments, color bar data representation

unpublished data

Acknowledgements

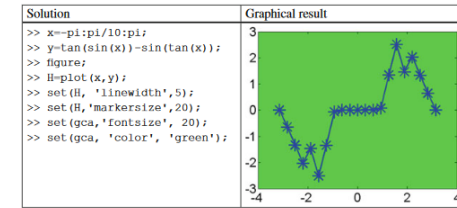
- Mauro Borgo, Alessandro Soranzo & Massimo Grassi (2012). MATLAB for psychologists, Springer New York.

<https://webdav.tuebingen.mpg.de/u/zli/MatlabForPsychologist.pdf>

- Take a brick to read this book for more tips
- Nicolas P. Rougier , Michael Droettboom & Philip E. Bourne (2014). Ten Simple Rules for Better Figures. PLoS Comput Biol 10(9): e1003833.

<https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003833>

- No joke, this article has a lot of good tips



A Brick for an Experiment

Plot the Results

MATLAB is a powerful tool for graphics. However, this brick requires a relatively simple graph. Usually, the results of experiments like that of Sekuler et al. (1997) are represented with bar graphs. Here too we will represent the results with a bar graph. We will draw a plot where the discs' motion (continuous versus with stop) is represented along the x-axis and bars are grouped by presence (or absence) of sound.

First, we need to get the means and the standard errors of the data we want to represent. We proceed as in the previous chapter. But first, we store the number of subjects we have run within the variable N.

Good Luck!

