MRC | Cognition and Brain Sciences Unit

Visualising data using Matlab

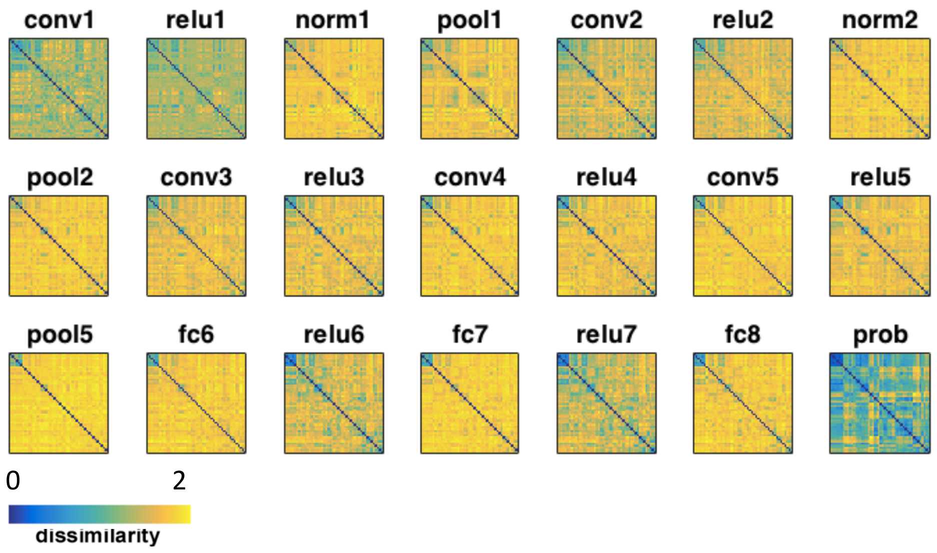
Kate Storrs

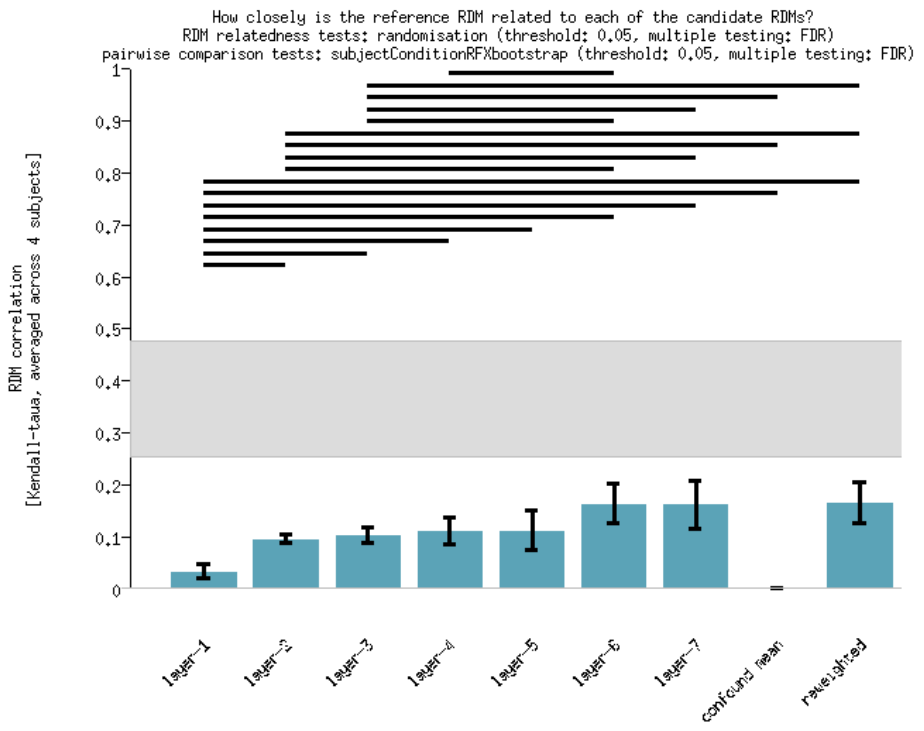
MRC Cognition and Brain Sciences Unit

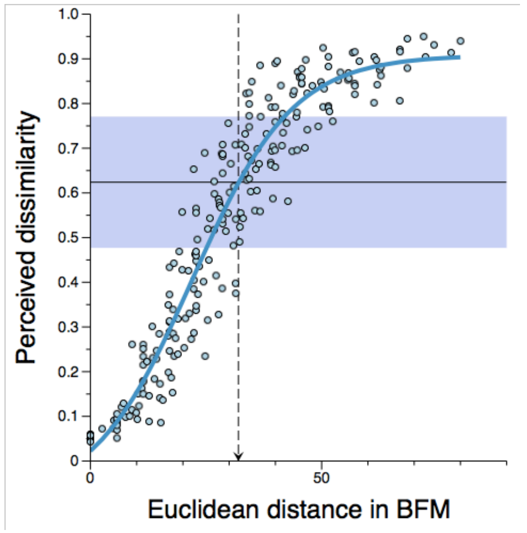
7th December 2016

quick survey: what do your data look like?

(some examples of what my data look like)



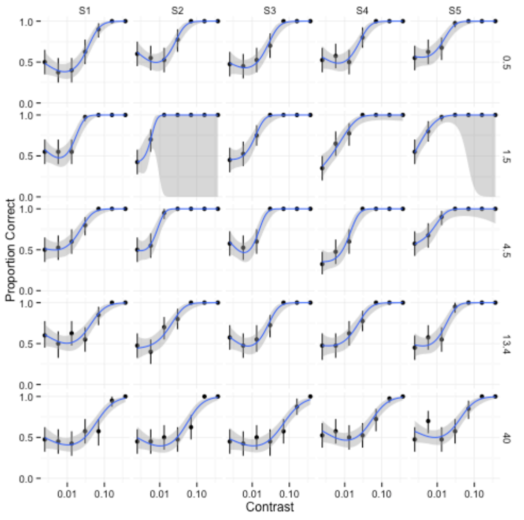




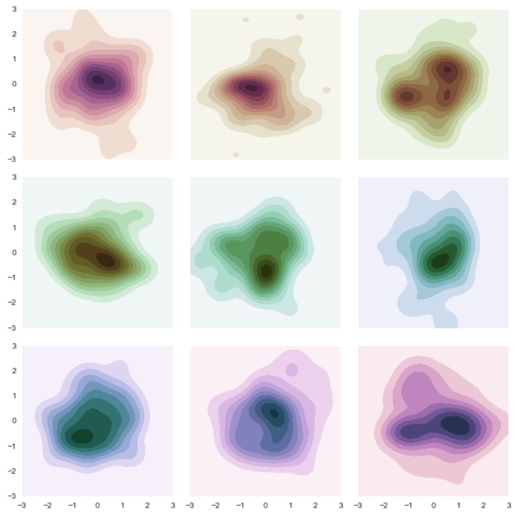
how to visualise data not using Matlab

Python / R

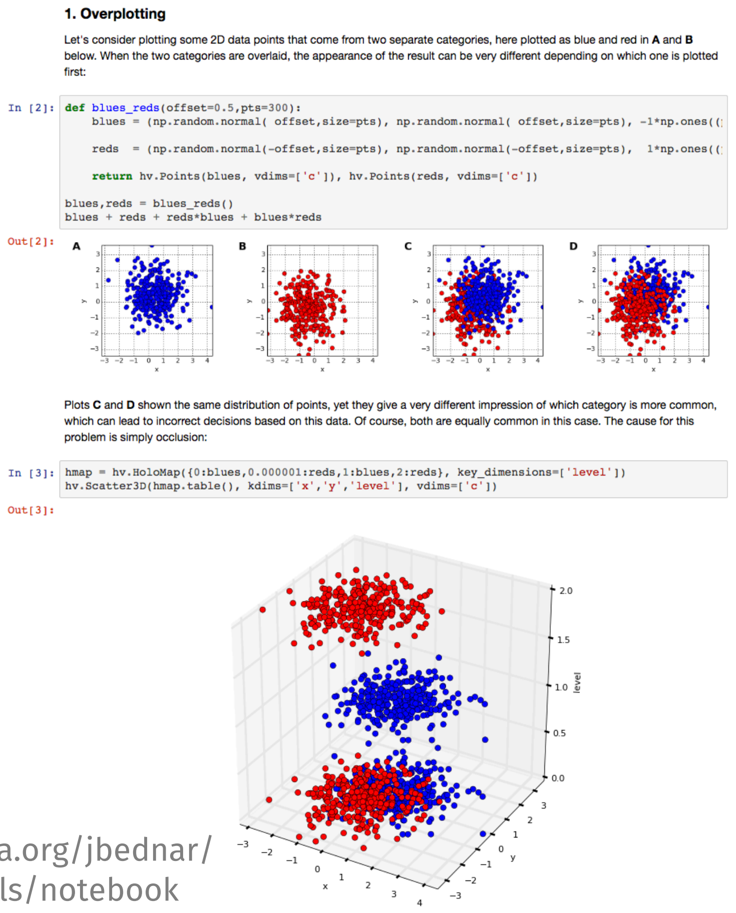
- free and open source software
- some very pretty visualisation toolboxes (e.g. ggplot2 in R, Seaborn in Python)
- interactive notebooks are great



https://tomwallis.info/2014/04/21/graphically-exploring-data-using-ggplot2/



http://seaborn.pydata.org/index.html



1. Overplotting

Let's consider plotting some 2D data points that come from two separate categories, here plotted as blue and red in **A** and **B** below. When the two categories are overlaid, the appearance of the result can be very different depending on which one is plotted first:

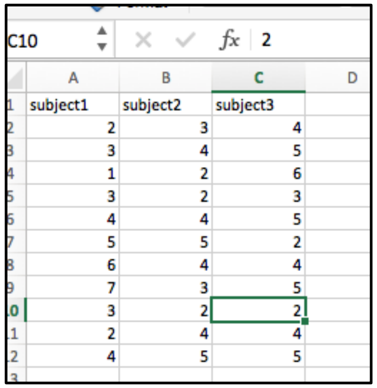
```
In [2]: def blues_reds(offset=0.5,pts=300):
            blues = (np.random.normal( offset,size=pts), np.random.normal( offset,size=pts), -1*np.ones((

            reds  = (np.random.normal(-offset,size=pts), np.random.normal(-offset,size=pts),  1*np.ones((

            return hv.Points(blues, vdims=['c']), hv.Points(reds, vdims=['c'])

        blues,reds = blues_reds()
        blues + reds + reds*blues + blues*reds
Out[2]:
```

Plots **C** and **D** shown the same distribution of points, yet they give a very different impression of which category is more common, which can lead to incorrect decisions based on this data. Of course, both are equally common in this case. The cause for this problem is simply occlusion:

```
In [3]: hmap = hv.HoloMap({0:blues,0.000001:reds,1:blues,2:reds}, key_dimensions=['level'])
        hv.Scatter3D(hmap.table(), kdims=['x','y','level'], vdims=['c'])
Out[3]:
```

https://anaconda.org/jbednar/plotting_pitfalls/notebook

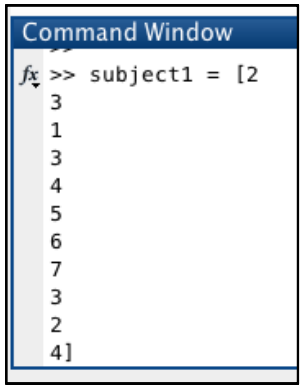
how not to visualise data using Matlab



C10 | fx 2

	A	B	C	D
1	subject1	subject2	subject3	
2	2	3	4	
3	3	4	5	
4	1	2	6	
5	3	2	3	
6	4	4	5	
7	5	5	2	
8	6	4	4	
9	7	3	5	
10	3	2	2	
11	2	4	4	
12	4	5	5	

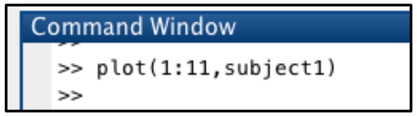
Type data into Excel…



Command Window

```
>> subject1 = [2
3
1
3
4
5
6
7
3
2
4]
```

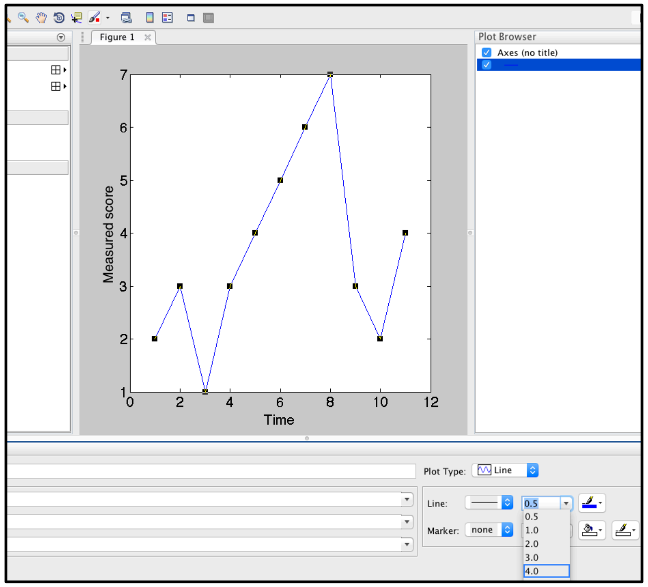
…copy-paste into Matlab…



Command Window

```
>> plot(1:11,subject1)
>>
```

…type in command line instructions to plot…



Plot Type: Line
Line: 0.5
Marker: none

…fiddle with plot using interactive plotting interface until satisfied.

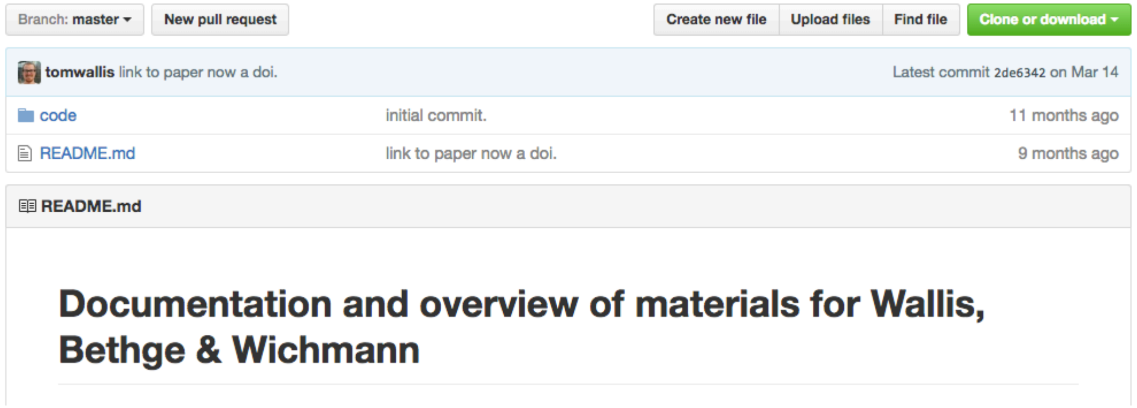
What's wrong with this?

- it's super tedious to do
- there's lots of room for human error
- it's hard to reproduce your figures (both for others and for future-you)

instead: good practice

In a Perfect Science World:

- When we're finished with a project, we have a succinct folder full of data files, analysis scripts, plotting scripts, and descriptions/instructions. Someone totally unfamiliar with our project can click a few buttons and get straight from our raw data to the figures and statistics in the paper/thesis chapter.
- cf. reproducible science (the 'carrot'). e.g.



tomwallis link to paper now a doi. Latest commit 2de6342 on Mar 14

code	initial commit.	11 months ago
README.md	link to paper now a doi.	9 months ago

README.md

Documentation and overview of materials for Wallis, Bethge & Wichmann

  - https://github.com/tomwallis/metamers_jov
- cf. the CBU Data Repository (the 'stick')
  - http://www.mrc-cbu.cam.ac.uk/wp-content/uploads/2016/09/Henson_CBUOpenScience_November2016.pdf

practical steps toward reproducible Matlab use

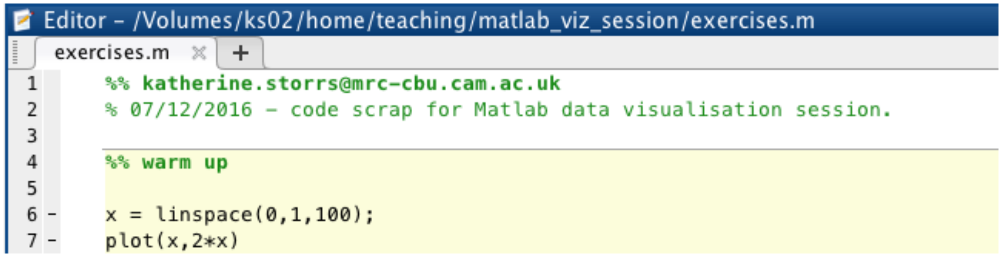
- Run plotting and analysis commands from scripts, not the command line.

- Load data, don't copy-paste or type it.
    - Likewise, automatically save outputs of experiments / analyses as .mat or .csv files for later re-use.
    - (generally no need to save figures as .fig files, as you should be able to regenerate at the click of a button)

- Curate your code, e.g.
    - Put a short description at the top of each script with your name, the date you created this script, and what it does.
    - Comment your code. For every line, if you're still new to Matlab.
    - If there's something you find yourself doing repeatedly, write it as a function in its own separate file.

warm up

Open a new script, with short description, and type 1-2 lines to make minimal plot


Editor - /Volumes/ks02/home/teaching/matlab_viz_session/exercises.m
exercises.m

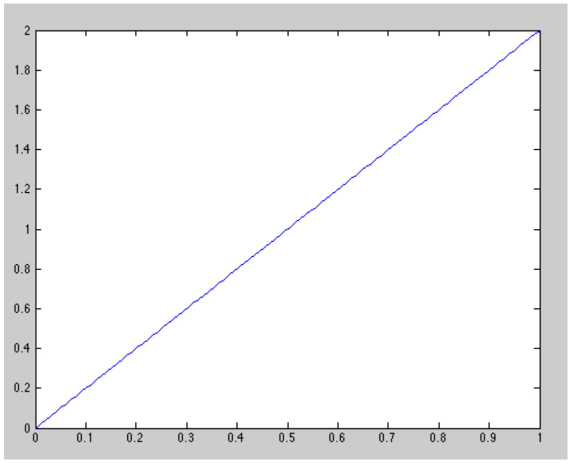
```
%% katherine.storrs@mrc-cbu.cam.ac.uk
% 07/12/2016 - code scrap for Matlab data visualisation session.

%% warm up

x = linspace(0,1,100);
plot(x,2*x)
```

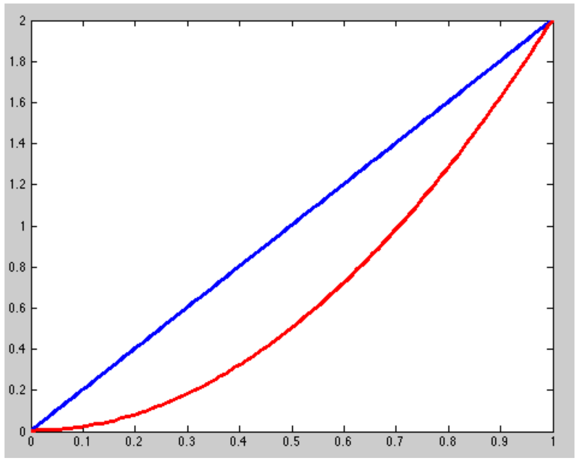
Type “help plot” in command line, and add code to change line style / colour

Type “hold on” under first plotting command. Add another plot in a different style

Use “xlabel”, “ylabel”, and “title” to add labels

Explore “box”, “axis”, and “legend” commands to make plot look ‘publishable’…





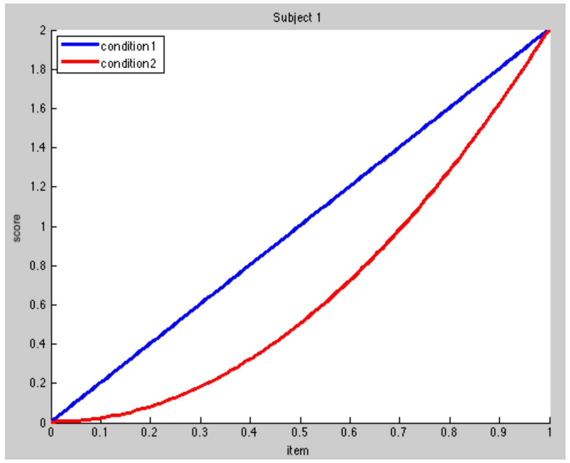
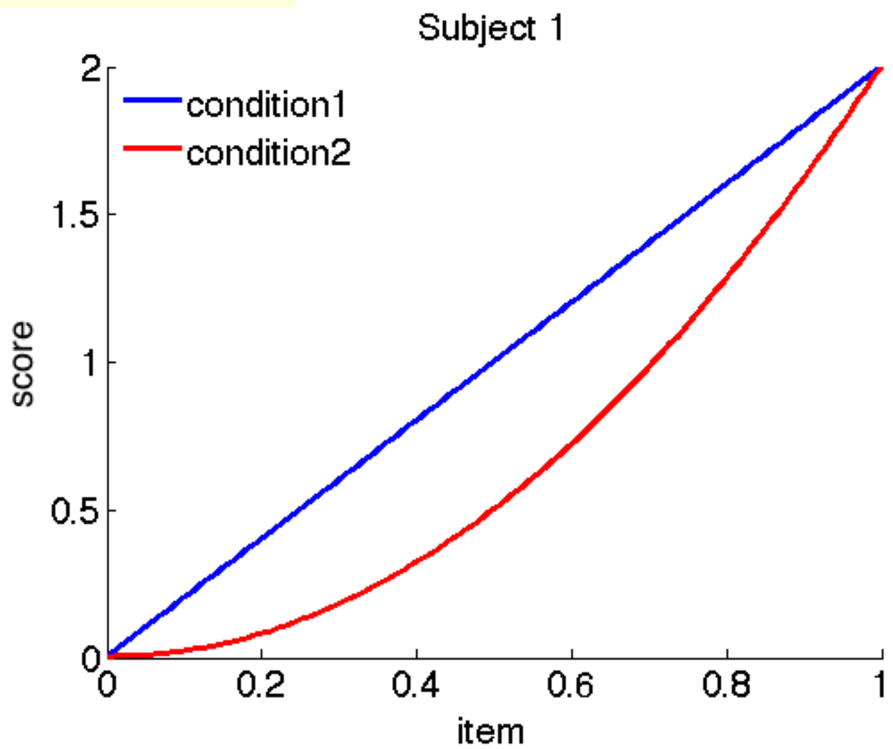


Figure and axis handles

Using “get current axis” and “get current figure” to set background colour, font size

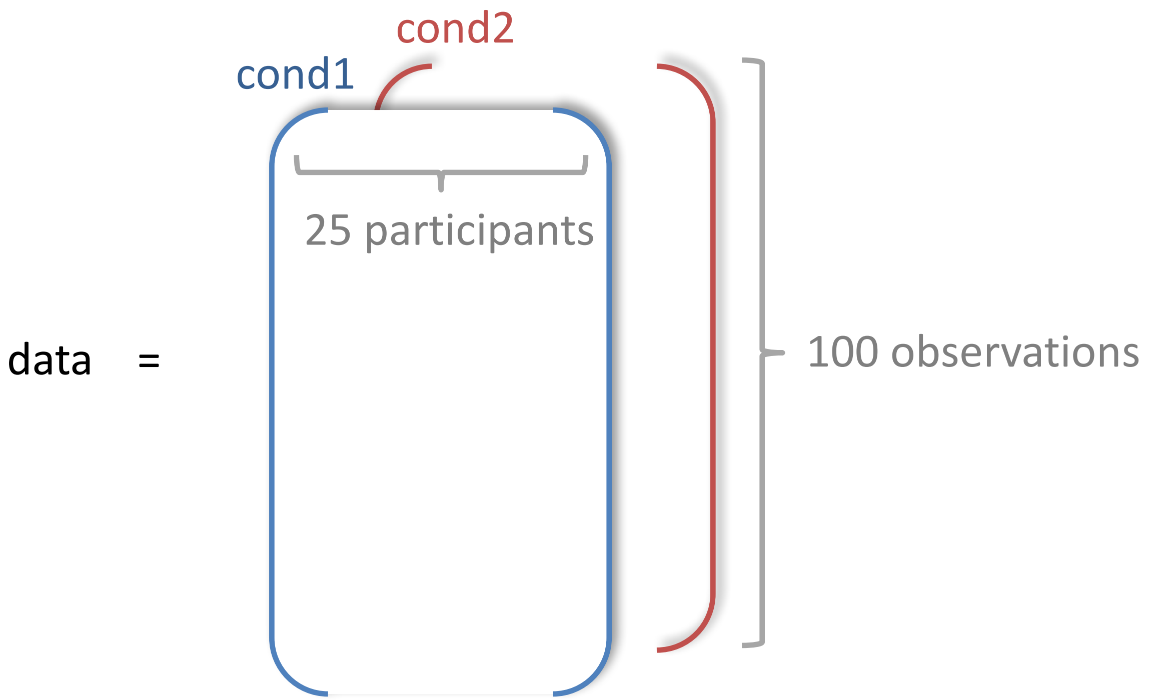
```
% plot two lines
x = linspace(0,1,100);
plot(x,2*x,'linewidth',3)
hold on
plot(x,2*x.^2,'r-','linewidth',3)
legend('condition1','condition2','Location','NorthWest')
legend boxoff

% adjust visual attributes
set(gca,'fontsize',20)
set(gcf,'color','w')
box off
title('Subject 1')
xlabel('item')
ylabel('score')
```



simulate and explore some data

Imagine 100 observations of some dependent variable, in two different conditions, for 25 participants.

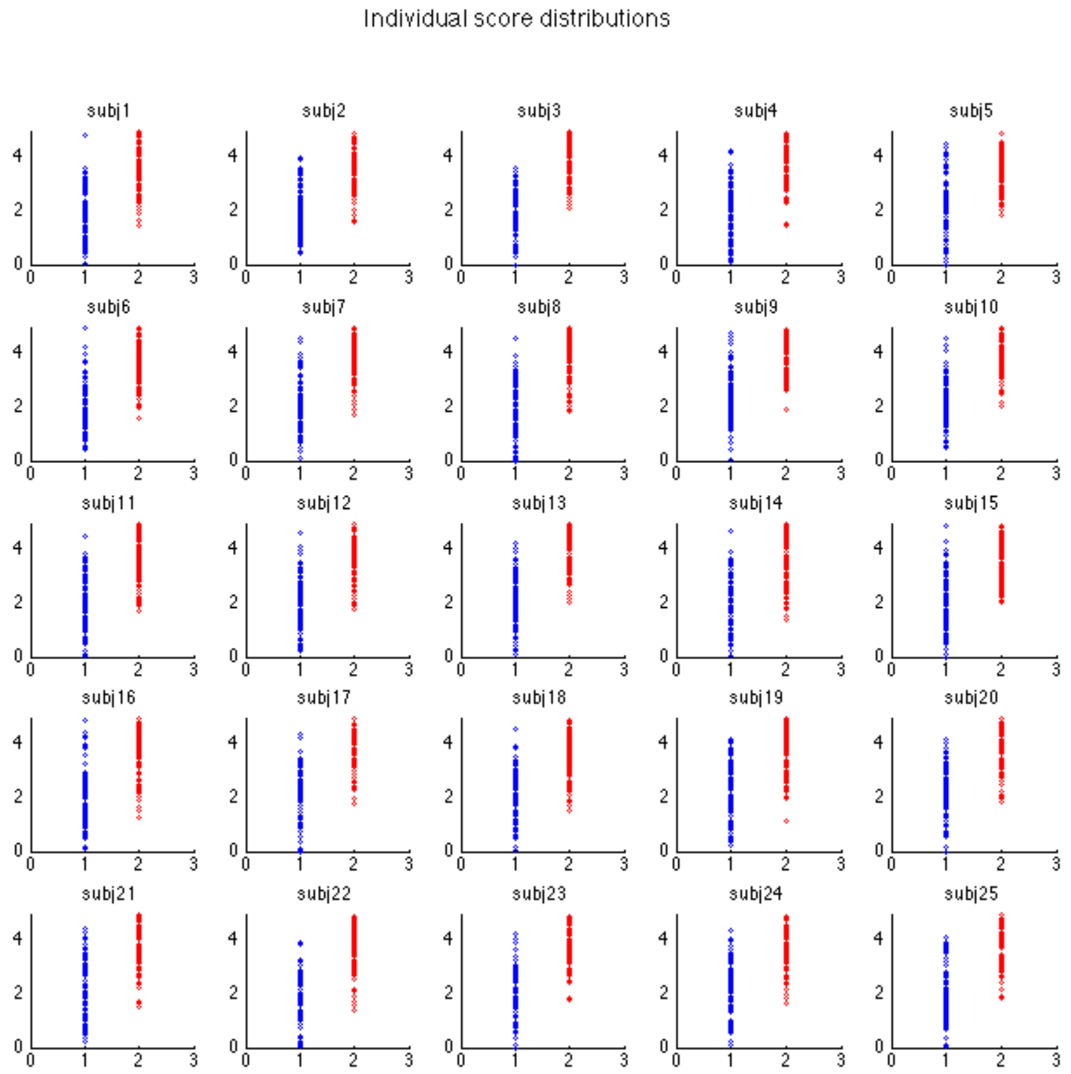
How would we structure our data from an experiment like this? How could we simulate it?



```
data(:,:,1) = random('norm',2,1,[100,25]);
data(:,:,2) = random('norm',4,1,[100,25]);
```

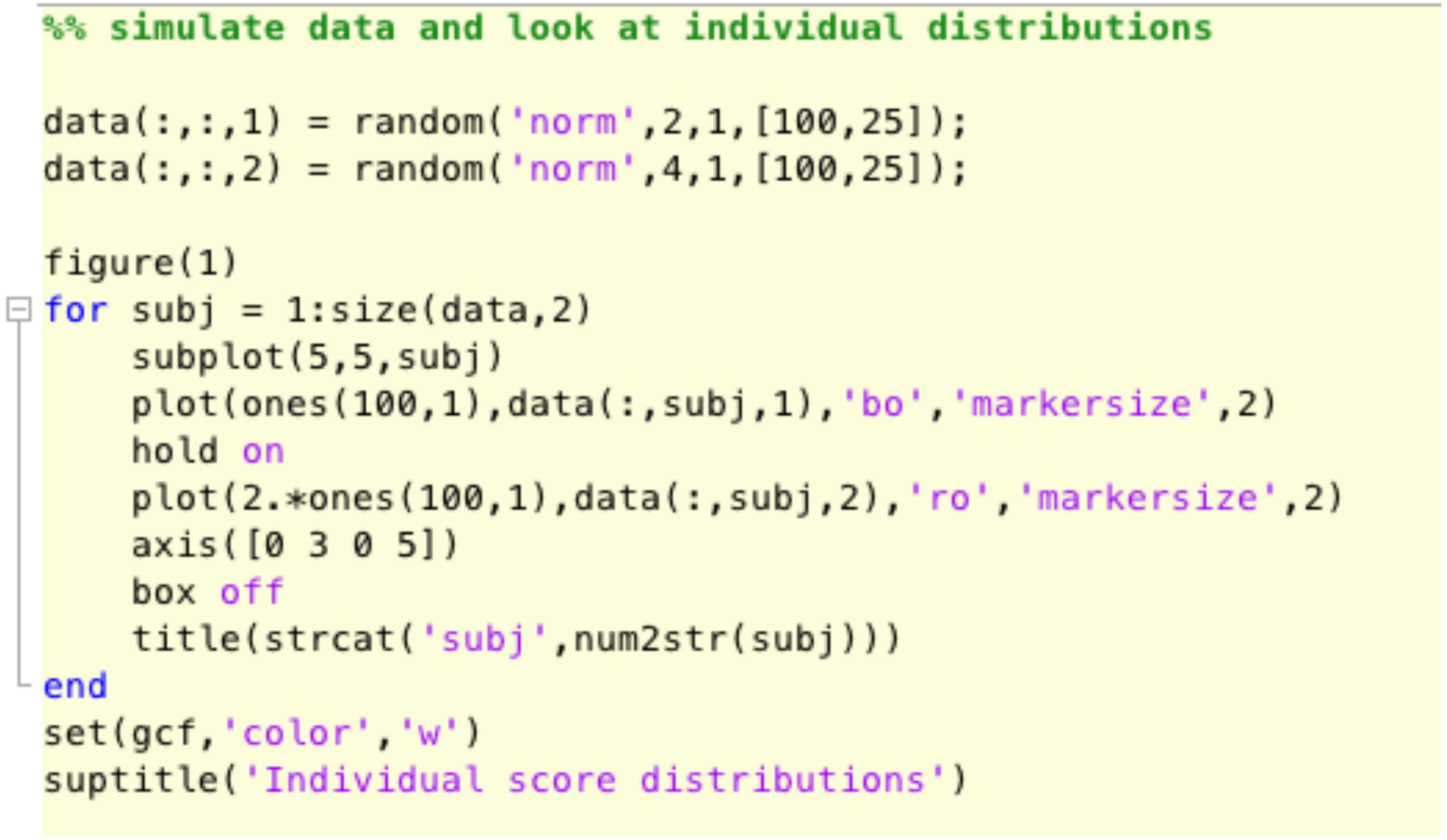
simulate and explore some data

How could we best look at our raw data for each individual participant? e.g.



simulate and explore some data

Example code...



```
%% simulate data and look at individual distributions

data(:,:,1) = random('norm',2,1,[100,25]);
data(:,:,2) = random('norm',4,1,[100,25]);

figure(1)
for subj = 1:size(data,2)
    subplot(5,5,subj)
    plot(ones(100,1),data(:,subj,1),'bo','markersize',2)
    hold on
    plot(2.*ones(100,1),data(:,subj,2),'ro','markersize',2)
    axis([0 3 0 5])
    box off
    title(strcat('subj',num2str(subj)))
end
set(gcf,'color','w')
suptitle('Individual score distributions')
```

simulate and explore some data

If we have time

How might we summarise and visualise these data across participants? e.g.

- Histogram
- Bar chart with error bars
- ...