MRC | Cognition and Brain Sciences Unit

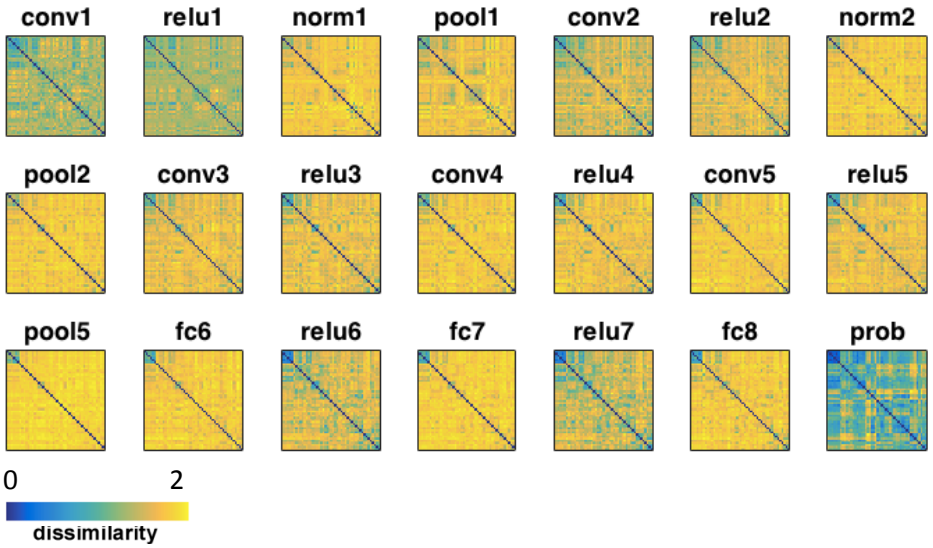
Visualising data using Matlab

Kate Storrs

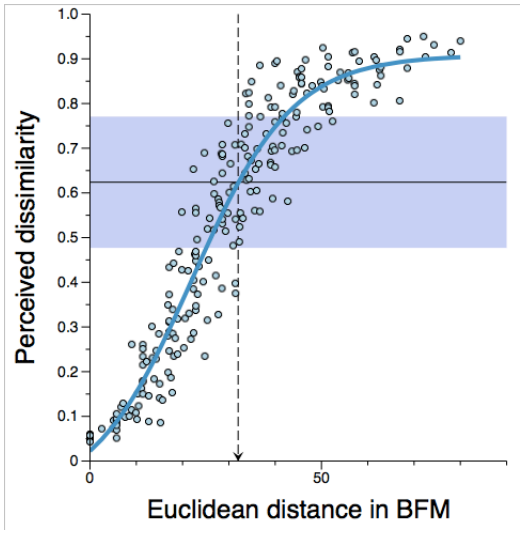
MRC Cognition and Brain Sciences Unit

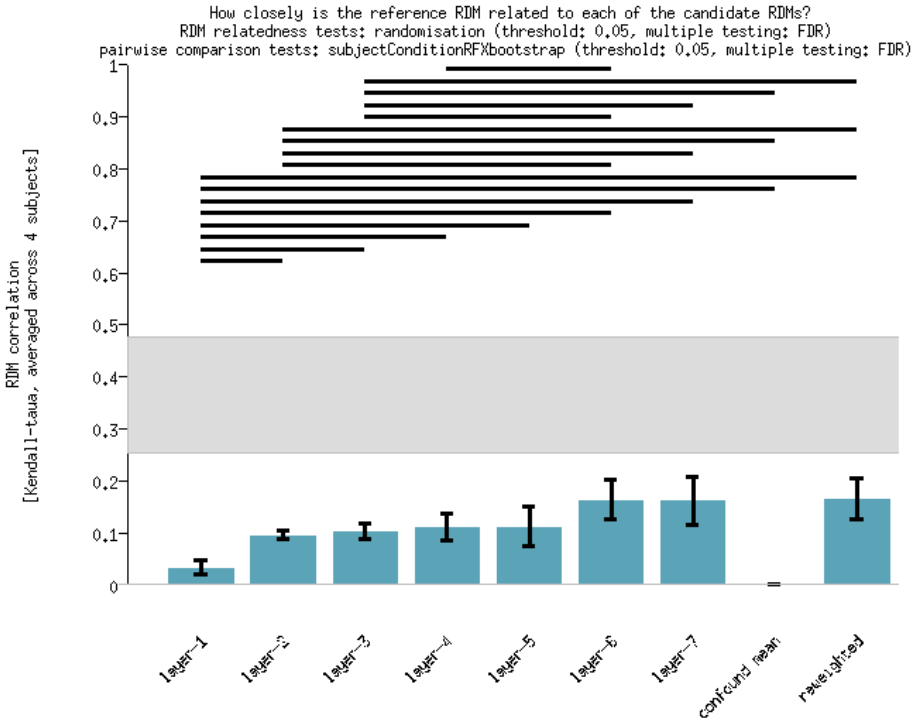
22nd November 2017

quick survey: what do your data look like?



(some examples of what my data look like)

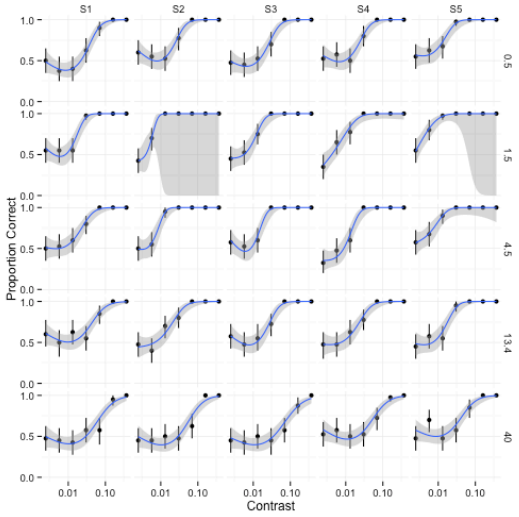




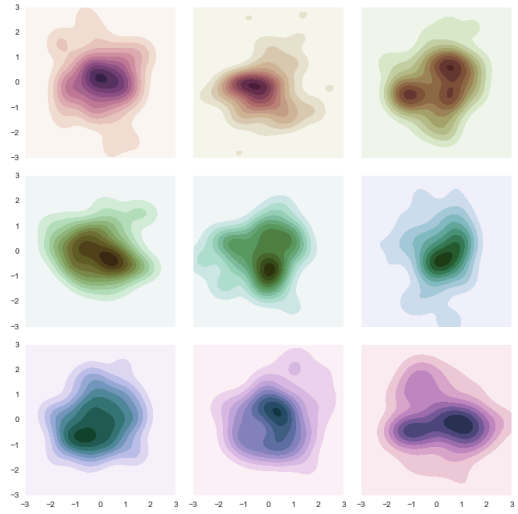
how to visualise data not using Matlab

Python / R

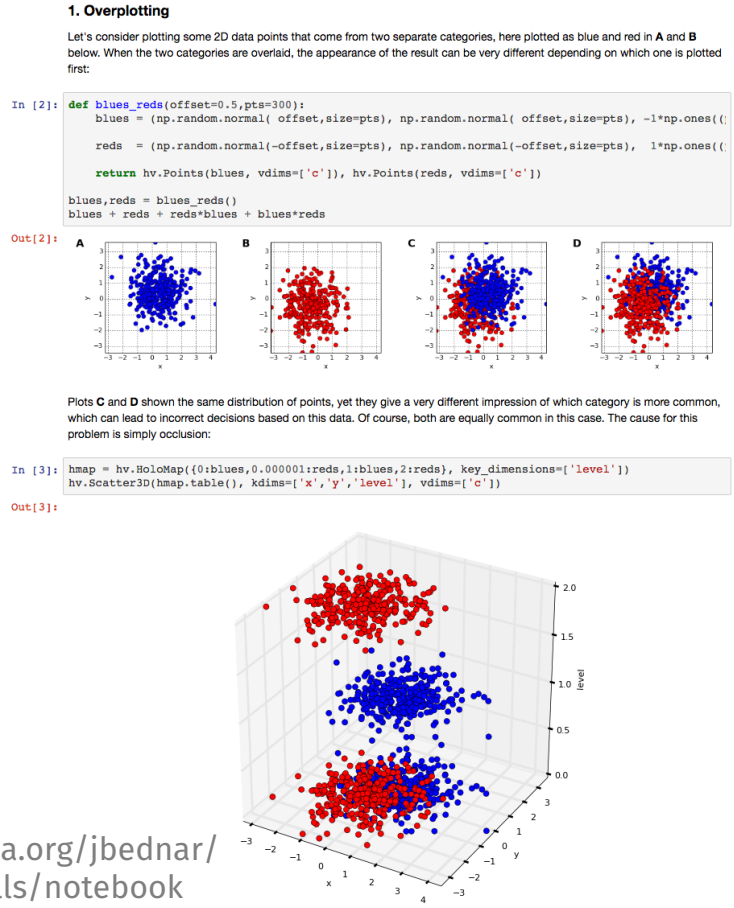
- free and open source software
- some very pretty visualisation toolboxes (e.g. ggplot2 in R, Seaborn in Python)
- interactive notebooks are great



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

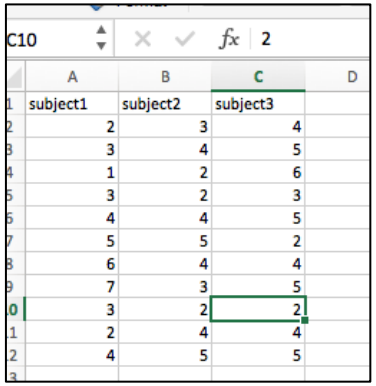


http://seaborn.pydata.org/index.html

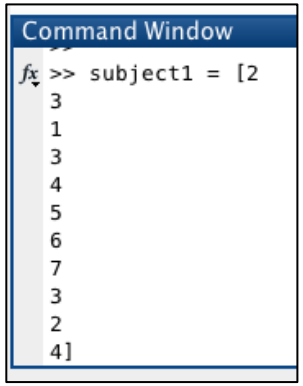


https://anaconda.org/jbednar/plotting_pitfalls/notebook

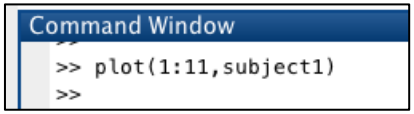
how not to visualise data using Matlab



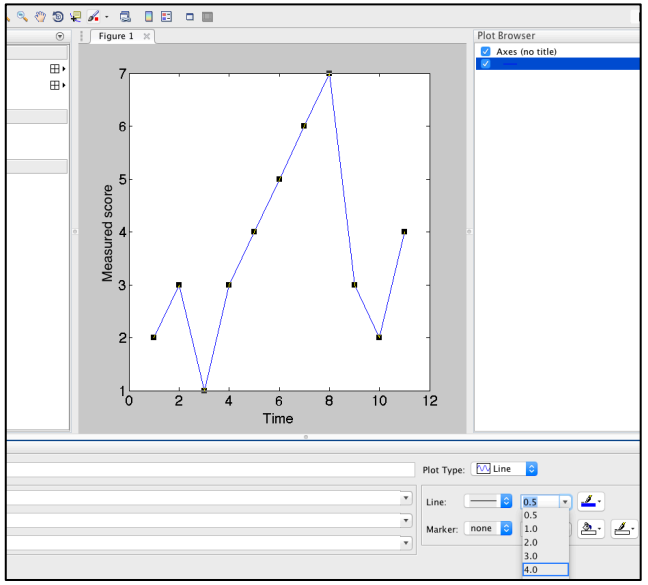
Type data into Excel…



…copy-paste into Matlab…



…type in command line instructions to plot…



…fiddle with plot using interactive plotting interface until satisfied.

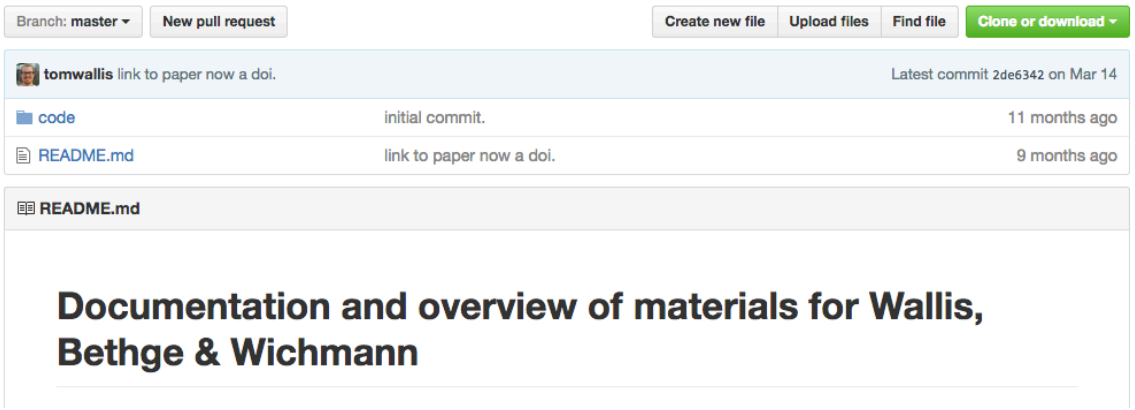
What's wrong with this?

- 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.



  - 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 it helps you follow old scripts.
  - If there's something you find yourself doing repeatedly, write it as a function in its own separate file.

exercise 1: warm up

Open a new script, with short description, and type 2-3 lines to make minimal plot

exercises.m

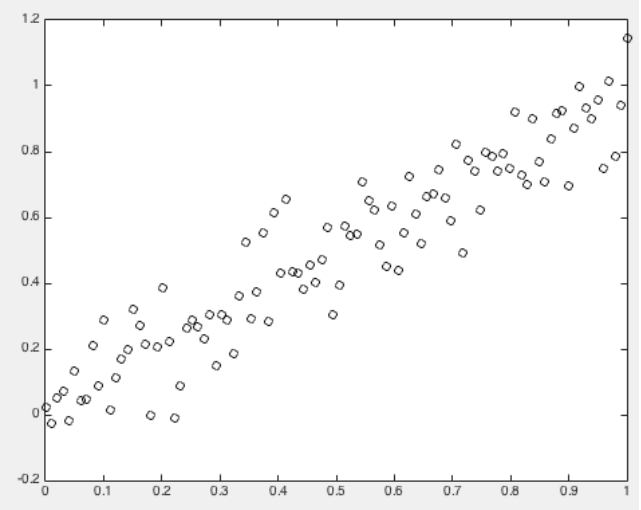
```
%% katherine.storrs@mrc-cbu.cam.ac.uk
% 22/11/2017 - example code for Matlab data visualisation session.

%% warm up

x = linspace(0,1,100);
y = x + random('norm',0,0.1,[1,100])
plot(x,y,'ko')
```

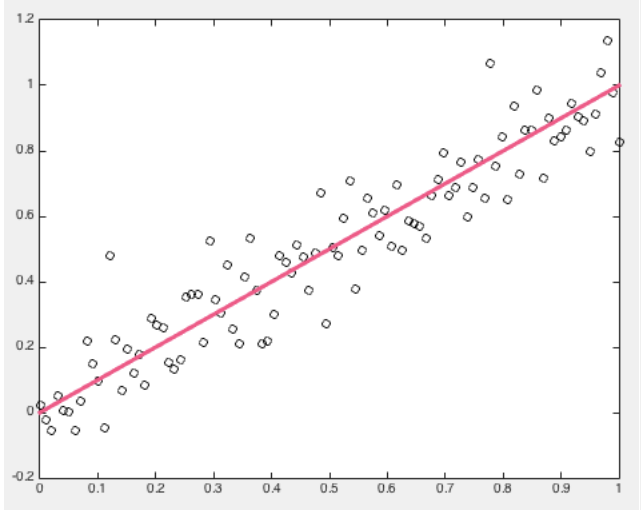
Type “help plot” in command line, and add options to change marker style/colour

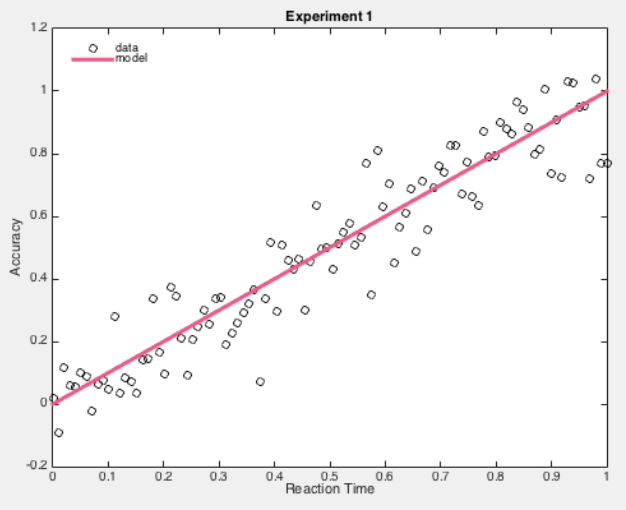
Type “hold on” under first plotting command. Add another plot in a different style

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

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





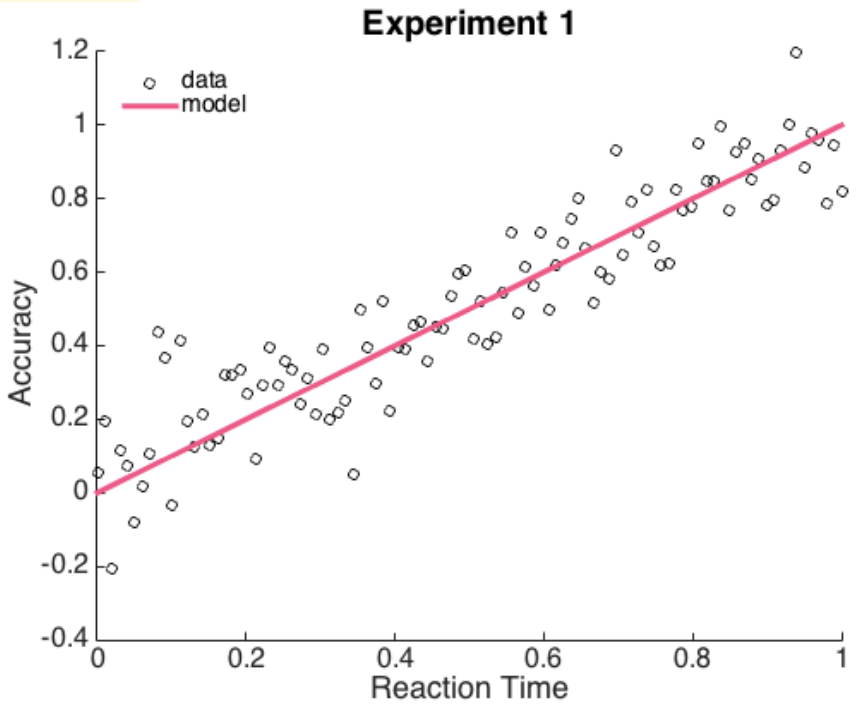


exercise 1: warm up

Figure and axis handles

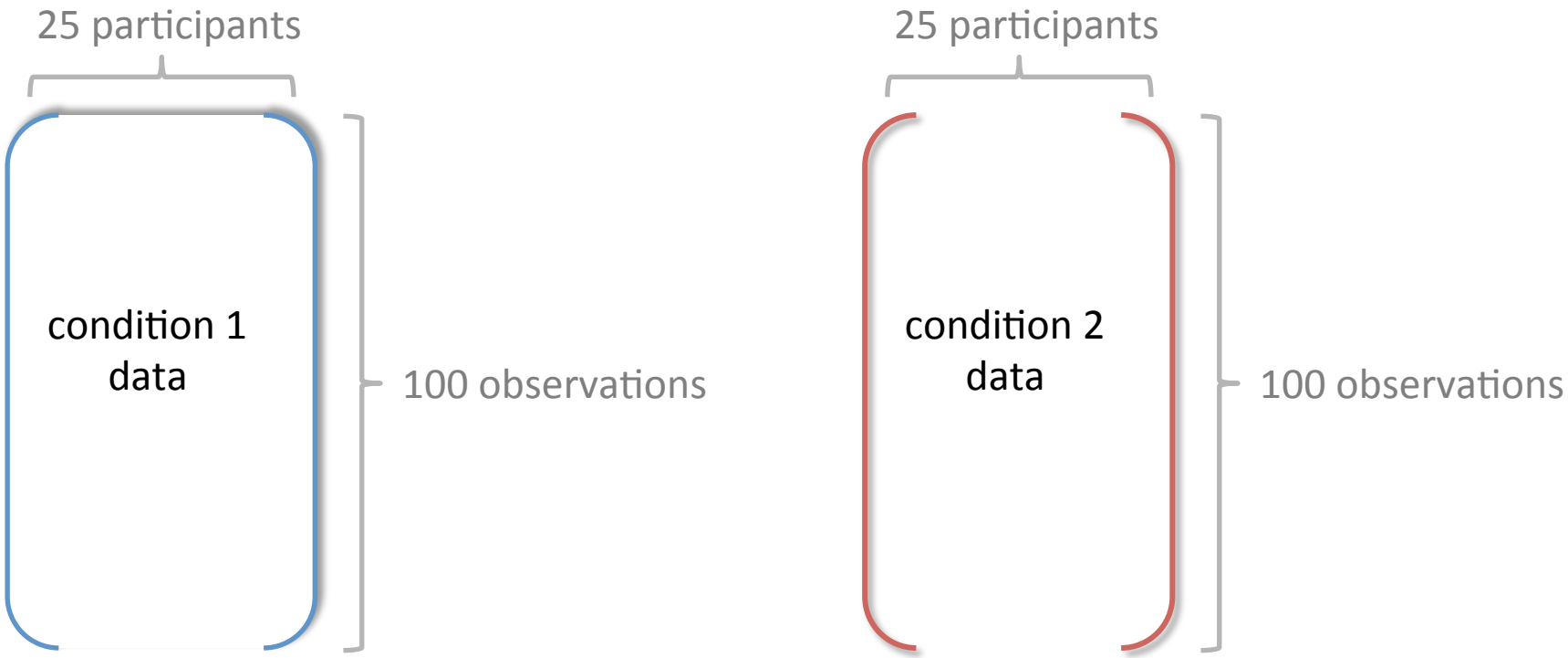
Using “get current axis” and “get current figure” to set background colour, font size

```
% plot dots and a line
x = linspace(0,1,100);
y = x + random('norm',0,0.1,[1,100])
plot(x,y,'ko')
hold on
plot(x,x,'-', 'color', [1, .2, .5], 'linewidth', 3)

title('Experiment 1', 'fontsize',20)
xlabel('Reaction Time')
ylabel('Accuracy')

legend('data','model','Location','NorthWest')
legend boxoff

% adjust appearance
set(gca,'fontsize',16)
set(gcf,'color','w')
box off
```



exercise 2: simulate and explore some data

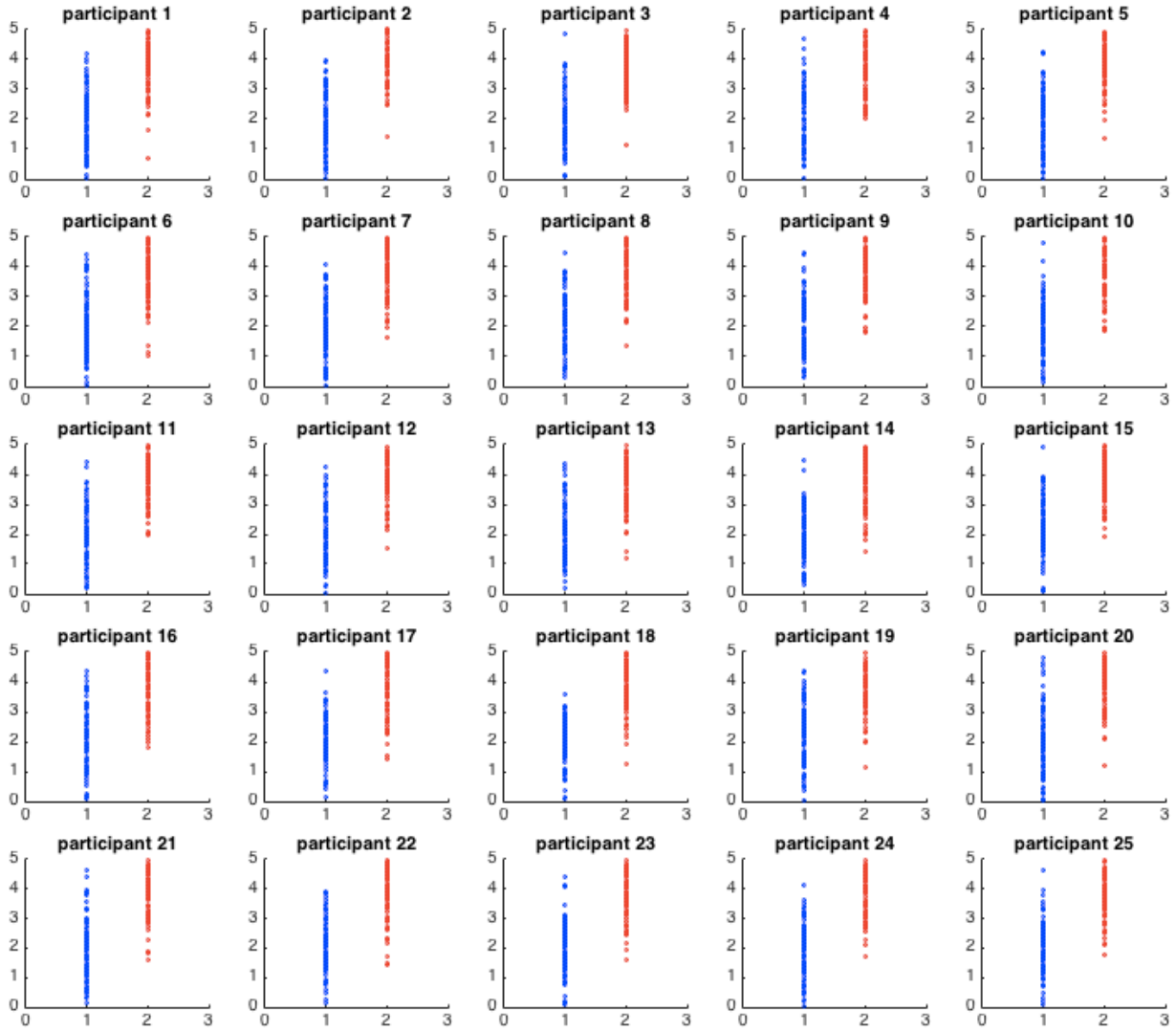
Imagine 25 participants each provide 100 ratings for items in two different conditions. How can we simulate and explore this dataset?




```
n_subjs = 25;
n_obs = 100;
cond1 = random('norm',2,1,[n_obs, n_subjs]);
cond2 = random('norm',3,1,[n_obs, n_subjs]);
```

exercise 2: simulate and explore some data

It's always good to first look at the raw data for each individual participant e.g.



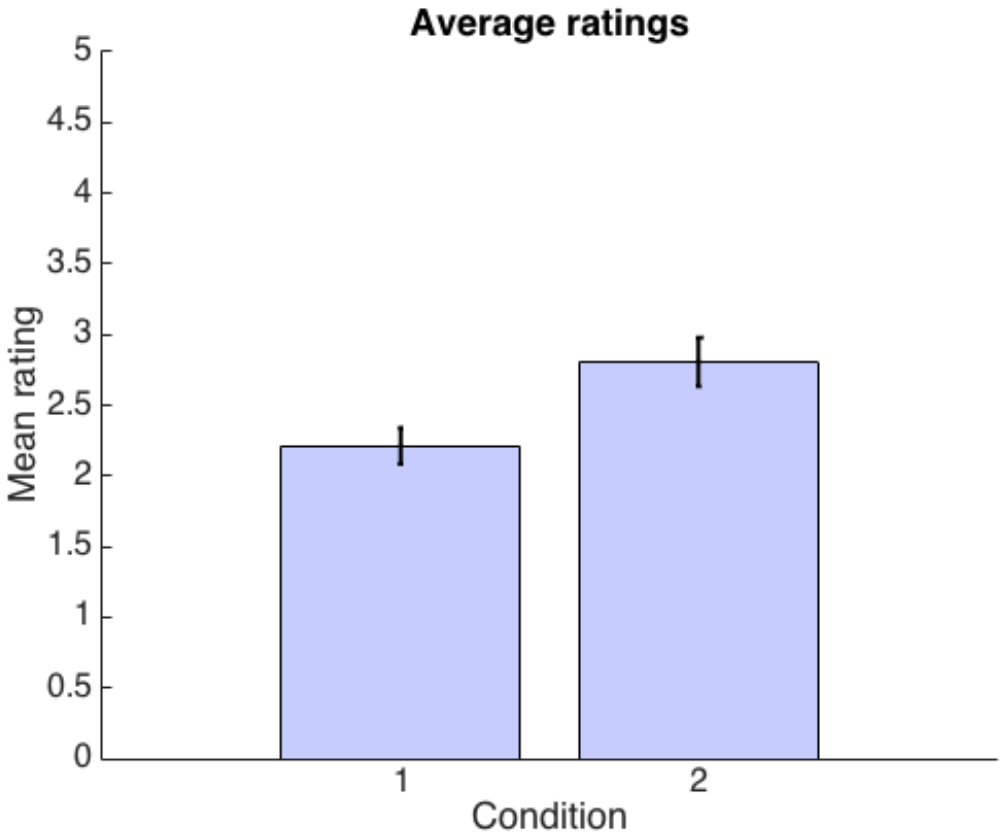
exercise 2: simulate and explore some data

Example code...

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

n_subjs = 25;
n_obs = 100;
cond1 = random('norm',2,1,[n_obs, n_subjs]);
cond2 = random('norm',3,1,[n_obs, n_subjs]);

figure(1)
for subj = 1:n_subjs
    subplot(5,5,subj)
    plot(ones(n_obs,1),cond1(:,subj),'bo','markersize',2)
    hold on
    plot(2.*ones(n_obs,1),cond2(:,subj),'ro','markersize',2)
    axis([0 3 0 5])
    box off
    title(sprintf('participant %1.0f',subj))
end
set(gcf,'color','w')
```

exercise 3: summarise the data

How might we summarise and visualise these data across participants? e.g bar char with error bars showing +/- 1 SEM across participants:




```
cond_means = [mean(mean(cond1)), mean(mean(cond2))];
cond_sems = [std(mean(cond1))/sqrt(n_subjs), std(mean(cond2))/sqrt(n_subjs)];
```

exercise 3: summarise the data

Example code...

```
%% summarise data

cond_means = [mean(mean(cond1)), mean(mean(cond2))];
cond_sems = [std(mean(cond1))/sqrt(n_subjs), std(mean(cond2))/sqrt(n_subjs)];

bar([1,2], cond_means, 'facecolor', [0.8, 0.8, 1])
hold on
errorbar([1,2], cond_means,cond_sems, 'k.', 'linewidth', 2)

xlabel('Condition')
ylabel('Mean rating')
title('Average ratings')

% adjust appearance
axis([0 3 0 5])

set(gca,'fontsize',16)
set(gcf,'color','w')
box off
```

Take-home messages

- Basic commands:
  - plot() for points, lines, curves, functions
  - bar() for bar charts
  - errorbar() to add error bars to points / lines / bars
  - subplot() to create multiple plots in one figure
  - title(), xlabel(), ylabel(), legend() to add information
  - set(gca, ...), set(gcf, ...) for finer control of appearance
- Best practice:
  - Load your data from a file, don't type or copy-paste it
  - Run your plotting commands from a script for reuse & replicability
- Most of programming is Googling!
  - lean heavily on Matlab Answers, StackExchange, File Exchange...