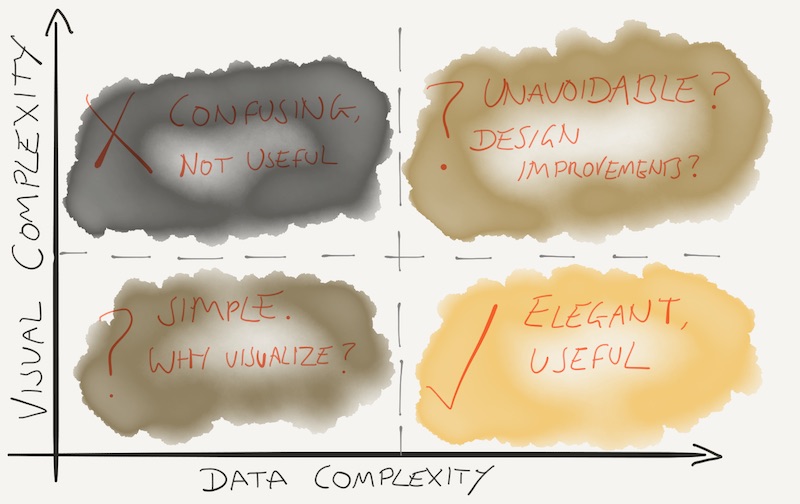
**Feedback on Session 1 Exercises**

**Tasks 1 and 2**

Thanks very much to those who have posted their assessments of data visualization examples from the [Information is Beautiful Awards](http://www.informationisbeautifulawards.com/showcase?acategory=data-visualization&award=2015&page=1&pcategory=long-list&type=awards). At the time of writing, 17 of the class of 80 have done this. *If you haven't put your evaluation up yet, can you please do so.* I would also encourage you to comment and respond to others' assessments. That way we can uncover some of the issues around data visualization design that should help you in creating your own graphics. The idea is that discussion board thread will become a useful resource for evaluations of good and poor data visualization design. It will be interesting too to compare your own evaluations now and at the end of this module. Assessing other data visualizations is a good way of developing your own visualization design skills and should help you in the weeks to come.

One theme that emerged in many of the visualizations chosen and your evaluations of them was the role that complexity plays the choice data and design of the graphics. I summerised it by considering data complexity and visual complexity as two independent elements:   
  
This is itself a simplification (e.g. sometimes there can be reason why a simple visual representation of simple dataset is desirable; sometimes making complex data look simple is too much of a simplification), but I hope it encourages you to think about what is gained and lost when transforming a dataset into something visual.

I hope also that everyone has viewed all of the [Journalism in the age of data](http://datajournalism.stanford.edu/) documentary. It gives a good sense of the motivations that encourage data visualization design and production as well as some of the trends in datavis for analysis and communication.

**Task 3**

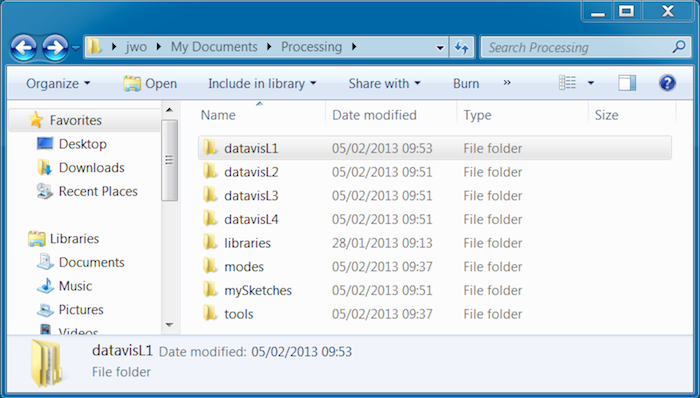
This task should have been straightforward - simply getting Processing running with a sketch copied from the lecture notes. If anyone still has problems installing Processing on their own machines, please post on the discussion board describing the problem. It is important that it is easy for you to create Processing sketches on your computer, so ironing out any problems with installing or running the software at this stage is important.

I saw plenty of examples of people experimenting with the sketches they had created or copied by changing small parts of the code and seeing what effect that had on the output. Please carry on doing this — this is exactly what Processing is designed to encourage.

A few people were getting into trouble when they tried to run two sketches from within Processing by creating a new 'tab' within their Processing coding window. We will be considering the use of tabs in a future session, but for now, if you want to run a second (or subsequent) sketch, you should do so by selecting from the File ->New menu (or keyboard shortcut Ctrl-N)/Cmd-N). This will open up a new window in which you can run an additional sketch without impacting on any existing open sketches.

You may find it handy to organise your sketches into a 'sketchbook' - that is a structured set of folders containing all your sketches. The default location for a sketchbook is My Documents/Processing (Windows) or Documents/Processing(MacOS X). If you create any folders inside this location, they will be directly accessible from within Processing. So, for example, you might wish to create folders in your sketchbook for each of these lectures.

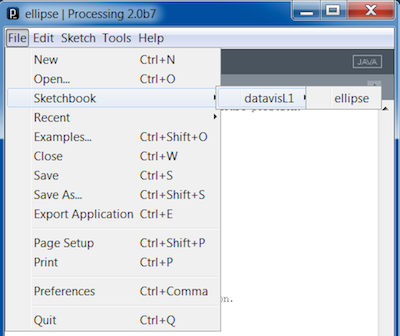
By default, Processing will name any new sketch with a slightly cryptic number representing the date you created the sketch. You should get into the habit of renaming your sketches as you save them with something that indicates what they do. This will help you when browsing your sketchbook as it begins to fill up.

[](http://staff.city.ac.uk/~jwo/datavis/session01/images/sketchbook1.png)

Sketchbook file locations (click to enlarge).

Any sketches placed in the sketchbook can be retrieved from the File->Sketchbook menu item:

If you are still having trouble getting Processing sketches up and running, or want to know a little bit more about the basics of creating sketches in Processing, have a look at the [Getting Started tutorial](http://processing.org/learning/gettingstarted/).

[](http://staff.city.ac.uk/~jwo/datavis/session01/images/sketchbook2.png)

Sketchbook view from within Processing (click to enlarge).

**Task 4**

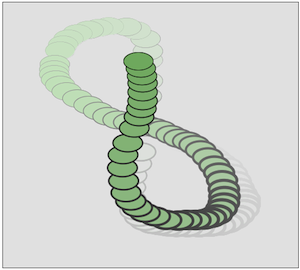
This task involved modifying the moving ellipse class to change its colour from red to green, have a longer 'tail' with movement, a different size that is proportional to the sketch size and its border to change in response to position. Here is my attempt to do this (click image to the right to see it in action):

|  |
| --- |
| // Draws an ellipse at the current mouse position.  // Jo Wood, 26th January, 2016    void setup()  {    size(500,450);    background(255);  }    void draw()  {    // Draw transparent background and rectangle with fixed border size.    fill(255,5);    strokeWeight(0.5);    rect(0,0,width-1,height-1);      // Draw ellipse at mouse position varying boundary by position.    strokeWeight(10.0\*mouseX\*mouseY/(width\*height));    fill(120,180,120);    ellipse(mouseX,mouseY,width/10,height/15);  } |

This should all have been possible with the techniques discussed in the lecture notes. Notice how the stroke weight is dependent on the mouse position (using the variables mouseX and mouseY). A few people who did this ended up with very thick strokes, so in my example I have divided by the sketch width and sketch height to scale it between 0 and 1, then multiplied by 10 to scale by 0 and 10 pixels wide.

A number of people noticed that when they moved the mouse, the entire window appeared to shrink. This was because the transparent rectangle that was drawn at the start if the draw() method also had a black border that changed stroke weight with the mouse. The strokeWeight() command, along with stroke() and fill() will affect all subsequent drawing, so unless you tell it otherwise, changing the stroke weight of the ellipse will also affect the stroke weight of the rectangle. You can think of setting the stroke weight, stroke colour and fill colour as selecting different 'pens'. At the stage the pens are selected, nothing actually happens until you draw something (such as a rectangle or ellipse). Any drawing will be carried out using the 'pen' you have previously selected.

To overcome this, simply set the stroke weight to a fixed value (I used 0.5 pixels in my example) before drawing the rectangle, then set it to some mouse-dependent value before drawing the ellipse. This represents an important way we have controlling the appearance of our sketches - *things are drawn in Processing in the order in which the commands in your sketch are written*. This allows us to draw features 'on top' of others by drawing them subsequently in our sketches.

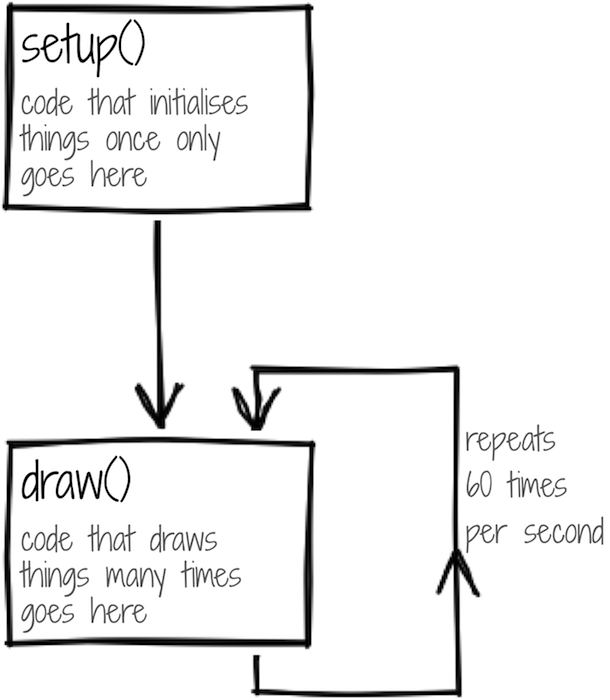
[](http://staff.city.ac.uk/~jwo/datavis/session01/practicalExercise4/javascript/index.html)

Moving ellipse with varying stroke weight (click to activate).

I noticed that some people were getting a little confused between what commands should be placed in setup() and what should be placed in draw(). Part of this confusion may have arisen because it is possible to place commands that draw things (like rect()) and commands that set the appearance of things (like strokeWeight() or fill()) inside both setup() and draw().

The difference between the two methods is that the code inside setup() will only ever be executed once and before anything inside draw() is executed. So things that need to be set up first such as loading data, or setting up a font for text display are usually placed inside setup(). You can also add stroke and fill settings here, *if you know they will not change within the lifetime* of the sketch. If you want such settings to change, such as the stroke width in my example above, they should be placed inside draw().

Note also that any stroke and fill settings that are in place by the 'bottom' of draw() will be remembered when Processing loops back up to the start of draw() on its 60-times-a-second round trip.

[](http://staff.city.ac.uk/~jwo/datavis/session01/images/processingFlow.png)

Processing flow (click to enlarge).

**Task 5**

The final task asked you to turn a table of opinion poll figures into a graphical representation using coloured circles. The purpose of the graphic was to illustrate possible trends changes in political support in recent years.

When presented with a design task like this, it can be a good idea to sketch out some possible ideas with pen and paper. This allows you to separate out the design issues from programming ones. This is particularly beneficial when you are still learning how to use Processing (or any other visualization environment) and the cost of coding is relatively high in comparison to the design stage.

In this case, we have 12 numbers to show representing 4 political parties over 3 years. Because we are interested in detecting trends, arranging the circles in rows or columns (as opposed to, for example, a circle) should help.

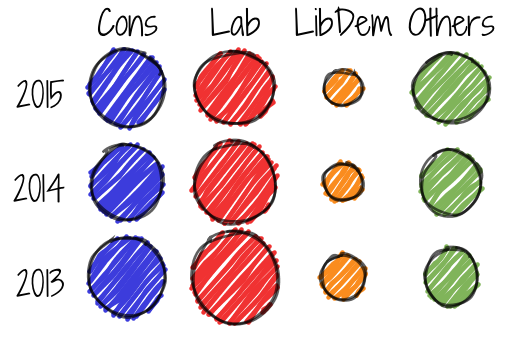
Using this grid arrangement we then need to show both party category (Labour, Conservative etc.) and the percentage support for each party. That's two 'attributes that will require two visual means to show each item. One obvious choice is to use colour to show party and size to reflect proportion of support for the party.

This layout and symbolisation makes it possible to see trends vertically within any party as well as the balance in support horizontally between parties.

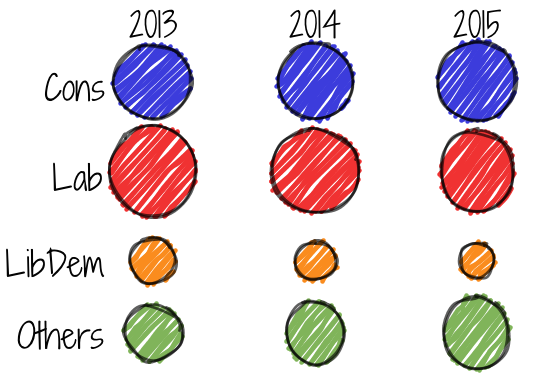
I noticed with some people interpreted trends as decreasing over time for 'others' and increasing over time for 'LibDem'. In fact the trend is the opposite, and the confusion arose because without labels, it may not have been obvious whether rows went forward or back in time from top to bottom. Are there ways of reducing the chances of such confusion? Although some people had found a way of displaying text in their sketches (see the [text() reference page](https://processing.org/reference/text_.html)), changing the layout to follow established convention can also help. So rather than arrange party from left-to-right and year from top-to-bottom, we can transpose the table so that time moves from left to right.

Such design refinements are common when creating data visualizations and you can save yourself much time by experimenting with them by sketching on paper before you begin coding.

My code to create a simple set of sized circles using the second gridded layout is shown below:

[](http://staff.city.ac.uk/~jwo/datavis/session01/images/votingPollSketch.png)

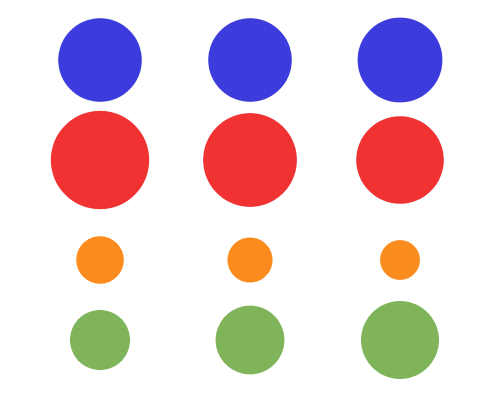
Gridded poll sketch design (click to enlarge).

[](http://staff.city.ac.uk/~jwo/datavis/session01/images/votingPollSketch2.png)

Alternative grid layout with time from left-to-right (click to enlarge).

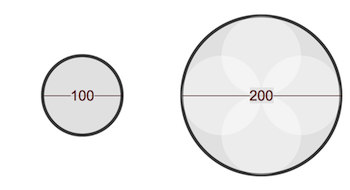
|  |
| --- |
| // Shows opinion polls Jan 2013 - Jan 2015  // Jo Wood, 26th January, 2016    // The voting data.  float con2015,con2014,con2013;  float lab2015,lab2014,lab2013;  float libdem2015,libdem2014,libdem2013;  float other2015, other2014, other2013;    void setup()  {    size(500,400);    float scaleFactor =width/32; // Scale circles relative to sketch size.      // Initialise the voting data and scale to fit in sketch.    con2015 = scaleFactor\*sqrt(32);    con2014 = scaleFactor\*sqrt(31);    con2013 = scaleFactor\*sqrt(31);    lab2015 = scaleFactor\*sqrt(34);    lab2014 = scaleFactor\*sqrt(39);    lab2013 = scaleFactor\*sqrt(43);    libdem2015 = scaleFactor\*sqrt(7);    libdem2014 = scaleFactor\*sqrt(9);    libdem2013 = scaleFactor\*sqrt(10);    other2015 = scaleFactor\*sqrt(27);    other2014 = scaleFactor\*sqrt(21);    other2013 = scaleFactor\*sqrt(16);  }    void draw()  {    background(255);   // White background.    noStroke();        // No borders around circles.      fill(60,60,220); // Conservatives in blue    ellipse(width\*.2,height\*.15,con2013,con2013);    ellipse(width\*.5,height\*.15,con2014,con2014);    ellipse(width\*.8,height\*.15,con2015,con2015);      fill(240,50,50); // Labour in red    ellipse(width\*.2,height\*.4,lab2013,lab2013);    ellipse(width\*.5,height\*.4,lab2014,lab2014);    ellipse(width\*.8,height\*.4,lab2015,lab2015);      fill(250,140,30); // LibDems in yellow    ellipse(width\*.2,height\*.65,libdem2013,libdem2013);    ellipse(width\*.5,height\*.65,libdem2014,libdem2014);    ellipse(width\*.8,height\*.65,libdem2015,libdem2015);      fill(128,180,90); // Others in green    ellipse(width\*.2,height\*.85,other2013,other2013);    ellipse(width\*.5,height\*.85,other2014,other2014);    ellipse(width\*.8,height\*.85,other2015,other2015);  } |

There are a few things to note in the Processing code used to produce this sketch. Each data item is stored in its own variable (we will see ways of doing this more efficiently for larger datasets in a later session). We can *declare* the variables to store all 12 polling numbers outside of both the setup() and draw() methods. By doing so, this means that both methods can have access to the same set of variables. We have *initialised* those 12 variables inside the setup() method and scaled each one by another variable - scaleFactor which we set to be a fixed proportion of the sketch's width.

[](http://staff.city.ac.uk/~jwo/datavis/session01/images/votingPoll.png)

Voting poll sketch.

Note also, and this is important, that the diameter of each circle is proportional to the *square root* of the number we are representing. This ensures that the *area* of the circle is proportional to the number it represents. If we had simply set the diameter to the number directly, we would have exaggerated larger numbers at the expense of smaller ones. For example, doubling the diameter of a circle from 100 pixels to 200 pixels actually increases its area by 4.

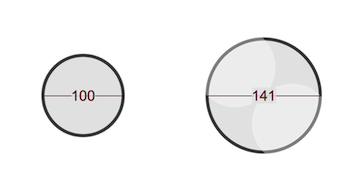
[](http://staff.city.ac.uk/~jwo/datavis/session01/images/circleSize1.png)

Doubling the diameter increases the area 4 times (click to enlarge).

If we wished to double the area in order to represent a data item that is twice as large as another, we need to increase the diameter by a factor of 1.41 (the square root of 2), not by 2.

Processing has a number of mathematical functions like [sqrt()](http://processing.org/reference/sqrt_.html" \t "_blank) to do this sort of processing of numbers. Have a look at the **Math** section of the [Processing Reference](http://processing.org/reference/) for more details.

The colours for each party set with the fill() command and were chosen to match their chosen political campaign colours, but with a slightly more pleasing less saturated look. You may find it easier to select the RGB colour numbers for any chosen colour by opening the **Colour Selector** from Processing's Tools menu. We will discuss colour in more detail in Session 3.

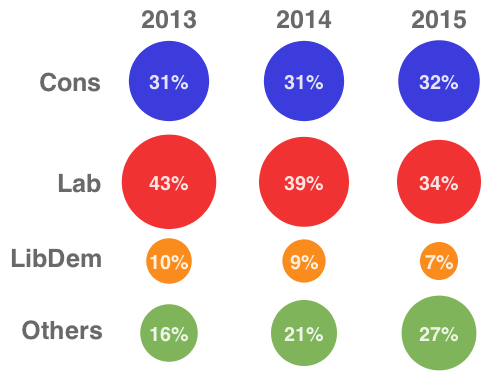
[](http://staff.city.ac.uk/~jwo/datavis/session01/images/circleSize2.png)

Increasing diameter by a factor of root 2 increases the area 2 times (click to enlarge).

This of course is only one way of answering the question. Some in the class had used layout effectively by overlaying the three circles for each party on top of each other. This made it easier to compare the differences between 2013, 2014 and 2015. It worked well in this case because for all four party categories there was a consistent trend over time. Others drew horizontal lines in rows around the circles. This enabled are more precise comparison between circle sizes. Someone else used a '100%' circle as a reference so that the circles representing 38.6%, 35.9% etc. could be compared with this one to gauge the proportion of votes. This was an interesting idea, although as we will see in later sessions, estimating the proportion that one circle takes up of another can be challenging.

Some people had managed to incorporate some text into their designs. While we hadn't covered this in the lecture materials of Session 1, it is fairly easy to experiment with text using the [Processing Reference](http://processing.org/reference/) and examples for guidance (but more on text in a later session). My attempt to add text to the figure is shown below:

The final part of the task asked you to consider how you might incorporate uncertainty in the polling results into your visual design. There was no requirement to code such a design, but I would be interested in your thoughts on how you might approach the design. In the lab session I only saw one example of such a design within the two hours, so please do post your thoughts on designing uncertainty into the visualization on the Moodle forum.

[](http://staff.city.ac.uk/~jwo/datavis/session01/images/votingPollLabel.png)

Design with text (click to enlarge).