# Assignment 3: Experiments in Biology - how to analyze tables of numbers

Due Thursday February 19, 5pm.

A great deal of biological data is stored in tables of numbers. For example, in microarray data you might store one row of information for each experiment, with each column either being a control variable, such as the presence of a stimulus or whether the cell or tissue is infected, or an output indicator, such as the degree to which a gene is expressed. Entries are generally either binary (yes or no), discrete (a small set of options), or continuous (can be normalized to the range (0.0, 1.0)).

There are so many ways to draw such data (see Chapter 7 of the book) that it is possible that no two members of the class will use the same method. The basic idea is to map each data row to a graphical object (point, line, polygon) and each value in the row to a graphical attribute (position, color, shape, size, ...). Some techniques allow you to use all dimensions at once, such as parallel coordinates and glyphs, while others choose subsets of dimensions for each plot (such as scatterplot matrices) or use dimensionality reduction methods to reduce N dimensions to 2 or 3 dimensions. The key is to make sure as much data as possible is scaled to fit on the screen while not leaving too much empty space. If you are using color to convey information, it is important to include a color key to help people interpret the data properly.

These are the minimum requirements.

*Remember that in the assignments part of your grade depends on going beyond the minimum requirements.* There are many opportunities to move beyond the basics. For example:

- Add ordering techniques that operate on the records or dimensions to reveal patterns of interest. Ordering can be done in many ways, including based on one of the dimensions, distance from a particular point, or relationships between dimensions or records.
- Show multiple views of the same data. Seeing data from multiple perspectives and using different visualization techniques can help a user form a better mental model of what's going on in the data. This technique becomes even more powerful if you allow interaction in one view to change the data displayed in the other views.
- Tackle a large dataset. Load the data into a database (likely hosted locally) and allow user interactions to drive the queries. Be wary of the overhead on queries, however. For example, if you're using a slider and you fire a query every time the user moves it by one pixel, your program will be slow.
- Use animation to show subsets of the data at a time. A cool effect can be achieved by having data appear at a time based on one of the dimensions

and then fade away as time progresses, similar to how some of you set up your simulation projects.

Don't restrict yourself to these examples. The goal is to develop and implement your own ideas.

## Resources

Matt Ward's course page has several links to datasets. Feel free to search elsewhere (Prof. Ryder's slides, for example).

The MultiVis.net site is a great resource if you want to explore alternative representations. Be sure to look at the top left for their other surveys.

If you use Sublime Text for coding, here is a tool that will allow you to run Processing from within your editor.

## Writeup Requirements

Your README.md should describe:

1. The dataset you use. It should be related to biology and included with the source code, or at least a portion of it so I can run the program. (Note: if you went the database route you will need to demo your project to me, but still submit the code.)
2. The major transformations you make to the data before visualizing it.
3. Whether your visualization can work with more than one dataset of the same type (e.g. more rows, but same dimensions).
4. If you are using outside code for the core of your project:
    - Identify the source(s) of the code.
    - Tell me how you modified it to make it your own work.
5. For the free-form (beyond the requirements) component:
    - Describe the biological significance of the additions you make.
    - Describe the technical significance of the additions you make.
6. Provide explicit instructions on running your program.

## Turning in the project

Submit the following:

- All of the source code

- A README file containing the elements listed in the Requirements section

The source code must run on a Mac or Linux machine.

Submissions should be made using myWPI. Either:

1. (Recommended) Upload text or a .txt file with a link to your GitHub repo, or
2. Zip your files and README and upload to MyWPI

## Grading

Each homework assignment is graded on a 100 point scale:

- 85 points will be based on whether the program fulfills the minimum requirements.
- 15 points will be based the additions you made:
  - 7.5 Bio-related additions
  - 7.5 Technical additions

Total – 100

(0 will be assigned if the code can't be compiled or run.)