Introduction

The purpose of this tutorial is to teach you the basics of loading, manipulating and visualising data in Processing with the HiVis library.

Outcomes

By the end of this tutorial you will be able to:

- load data from a spreadsheet with HiVis;
- perform basic manipulation or “shaping” of the data with HiVis; and
- visualise the data using basic drawing commands in Processing.

Getting started

The first thing to do is install HiVis and another library, Beads, that we’ll use to “render” some data with sound.

1. Open Processing.
2. In the menus go to Sketch > Import Library > Add Library
3. In the filter box type in “hivis”
4. Click on the HiVis library and then click the Install button.
5. Repeat steps 3 and 4 for the Beads library.

Loading data from spreadsheets / Terminology

HiVis expects the data in a spreadsheet to be in a particular format. Each row of the spreadsheet should contain one record or “data point”. For example if the data is recording the name, hair colour and height of people then the spreadsheet will have columns for name, eye colour and height. The first row may optionally contain headings for the columns. Thus the spreadsheet will look something like this:

Name	Eye colour	Height
Fin	Brown	161
Tovar	Green	183

When HiVis loads this spreadsheet it will create a *DataTable*. The DataTable will consist of three *DataSeries*, one for each column. The series will be of length 2 as the first row will be used to label the series in the table. When manipulating the data we'll be able to refer to the series by their index in the spreadsheet or by their label (if present in the spreadsheet).

A note on indexing into DataTables and DataSeries: when referring to series in a table or an element of data in a series by their index, the indexes start at **0**, not 1. So to access the first series in a table you might write something like `myTable.get(0)`, or to access the third element in the second series: `myTable.get(1).get(2)`.

Exercise 1 – Scatter Plot

In the first exercise we'll look at producing a scatter plot. In this plot we draw a dot for each data point of the *Iris* dataset (the data is in the *iris.csv* file in the same folder as the tutorial sketch). The position of a dot (the x and y coordinates) is determined by two values from the data point.

1. In the Processing menus go to *File* > *Examples*. Then open *Contributed Libraries* > *HiVis* > *tutorials* > *T01_Scatter*.
2. Click Run. You should see a scatter plot.
3. Open the *iris.csv* file in a spreadsheet program (eg Excel or Calc). Try changing the values in the spreadsheet and saving it. The plot should be updated with the new values. Now try adding new rows (data points) in the spreadsheet and saving it.
4. Close the plot.
5. Take a moment to read through and get familiar with the code. Don't worry if some new commands or methods don't make sense immediately.
6. The plot is using the *Sepal.Length* and *Sepal.Width* columns/series for the *x* and *y* coordinates respectively. Change it to use different series for the coordinates and run it again.
7. The plot is only representing two values from each data point. Let's change it so that it can represent three values. We'll do this by changing the brightness of the dots according to the values in one of the series:
    a) Add a new variable to specify the series to use for the brightness (see lines 17 and 18).
    b) In the `fileSelected` method we create a DataTable called `scaledData` containing scaled views of the series for the *x* and *y* coordinates. Add another series to this table that is a scaled view of the brightness series (the existing scaled views are labelled "*x*" and "*y*" in

the scaledData table, call the new one *"b"*). Set the scaled series view to have range [63, 255]. Note that the the `DataSeries.toRange` method accepts two values: the minimum and maximum values to scale the data in the original series to.

c) In the `for` loop in the `draw` method extract the value from the scaled version of the brightness series into a new `int` variable to store the brightness for the dot. We get the value as an integer as we're going to use it as an argument to the fill command. Hint: you'll need to use the `getInt` method.

d) Add a `fill` command just before the `ellipse` command to change the brightness of the dot, using the new brightness variable you added.

Exercise 2 – Pie Charts and Data Manipulation

Let's make some pie charts based on some statisticatical calculations over the data in the *Iris* dataset.

1. In the Processing menus go to *File > Examples* (if it's not open already). Then open *Contributed Libraries > HiVis > tutorials > T02_Pie*.
2. Take a moment to read through and get familiar with the code.
3. Click Run. You should see 3 pie charts.
4. The last pie chart shows the relative ratios of the average petal length and width of each species. Let's modify the code to display a fourth pie chart showing the relative (approximate) area of the petals for each species:
5. Copy the four lines of code under the comment `// Draw a pie chart showing the average ratio of petal length to width for each species` and paste them directly underneath those lines.
6. In the code you just pasted, change the ratio calculation `petalLength.divide(petalWidth)` to an ellipse area calculation $\frac{length}{2} \times \frac{width}{2} \times \pi$: `(petalLength.divide(2)).multiply(petalWidth.divide(2)).multiply(PI)`. Note that just calculating the length multipled by the width would produce an identical pie chart, but this demonstrates how to create complex DataSeries operations.
7. Change the text for the label under the new pie chart.
8. Click Run and see how it looks.

Exercise 3 – Sonification

In the third exercise we're back to the scatter plot, but this time with a twist..

1. In the Processing menus go to *File > Examples* (if it's not open already). Then open *Contributed Libraries > HiVis > tutorials > T03_Sonification*.
2. Take a moment to read through and get familiar with the code.
3. With the volume turned up on your computer, click Run and then try hovering the mouse pointer over the dots. Irritate your friends by composing a tune with your deft hovering skills.
4. I wonder which dots belong to which species? Would including the species in the visualisation show any patterns? Let's find out.
5. Under the line `DataSeries freqSeries;` add the line `DataSeries hueSeries;`
6. Under the line `freqSeries = data.get(2).toUnitRange();` add the lines (you might like to try copying and pasting):
7. ```
   hueSeries = data.get(5).asString().apply(new Function() {
     public Object apply(String species) {
       if (species.equals("setosa")) return 0.0;
       if (species.equals("virginica")) return 0.33;
       return 0.67;
     }
   });
   ```
8. Finally, change the line `float hue = 0;` to `float hue = hueSeries.getFloat(row);`
9. Click Run. Bask in the clarity.
10. Play around with the sonification. You could try changing the volume based on one of the properties. You could try changing the wave form baed on the species (other than SINE, there's SAW, SQUARE, TRIANGLE, and NOISE).
   ```