Eurographics Conference on Visualization (EuroVis), Posters Track (2017)
A. Puig Puig and T. Isenberg (Editors)

Visualising Data with L2

Graham McNeill

Oxford Internet Institute, University of Oxford, UK

Abstract

L2 is a new compile-to-JavaScript language for data wrangling, analysis and visualisation. The language is used in its own interactive environment (desktop or browser-based); compiled code will run in any browser or Node.js. To visualise data, L2 uses high-level libraries for simple plots, but also makes it easy to create SVG or canvas-based visualisations from scratch. Here, we discuss the various approaches to visualising data in L2 and consider a simple example.

1. The L2 Language and Environment

L2 is a new compile-to-JavaScript language for data wrangling, analysis and visualisation. The language can be used interactively in the L2 Environment which runs as a desktop application (using NW.js [nwj], Figure 1) or in the browser and includes a console, editor, plot panel, tutorial and documentation. Compiled L2 code is standard ES5 JavaScript that will run in Node.js or in any browser. L2 tables and plots are easily displayed in HTML documents and L2 includes powerful HTML operators that can be used for general client-side development and creating custom visualisations.

The L2 language is very simple; it has eight types and a single data structure called a *table*—a 3-dimensional array whose rows, columns and pages can be referred to using indices and/or keys. Operators (built-in functions) act on tables in an intuitive way, similar to array-oriented languages such as Matlab, R and J. When writing code, infix notation is used at all times (the operator goes after the first argument) and only parentheses have high precedence. These straightforward rules, combined with Python-like indentation-based blocks make it easy to read and write code.

2. Visualisation

For simple plots, L2 uses popular, high-level libraries. For more complex tasks, it is easy to create SVG or canvas-based visualisations from scratch.

Flot: Plot operators use the Flot library [Lau]. Flot is fully incorporated into the L2 language and no knowledge of the library is required. Plot operators can be passed a standard 'data matrix' or in more complex cases, such as data series of different lengths, *boxed data* can be used—a *box* is the L2 type used to put tables inside tables. Boxed data can be plotted on a single set of axes or as a trellis plot.

Vega and Vega-Lite: The well-known Vega [SWH14] and Vega-Lite [SMWH17] libraries create a visualisation from a JSON 'specification'; a single structure that contains both the data (or its url) and instructions for visualising it. In L2, we represent the specification as a table with row keys (which is very similar to a JavaScript object), manipulate it using standard L2 operators and call the `.vgPlot` or `.vgLite` operator to create the visualisation. L2 includes various operators for working with JSON data and boxed data more generally. There are also operators for converting between boxed (hierarchical) data and 'flat data'. All functionality of Vega and Vega-Lite is available in L2, but since we interact with the Vega specification directly, knowledge of the libraries is required.

HTML: The table-based nature of L2 makes DOM manipulation significantly easier and more compact than with JavaScript or jQuery. This is demonstrated in Figure 1 where table-based string concatenation is used to generate multiple HTML elements in a single expression (line 14). On line 35 in Figure 1, we set various attributes of circle elements (a different value for each element) in a single expression. Flot and Vega can be used in HTML documents. The same syntax is used as in the L2 Environment, except that we also specify the HTML element where the plot is to be displayed (lines 19 and 25). Similarly, L2 tables are easily displayed in HTML documents. Tables are styled using standard CSS and table cells are automatically given intuitive class names and IDs so that individual cells and sets of cells are easily selected.

Canvas: L2 uses p5.js [McC] for drawing on the HTML canvas. p5.js makes it easy to draw 2 or 3-dimensional 'sketches' which can be static or animated. L2's table-based sketch operators make sketching simpler still, particularly for data visualisation where verbose code including many loops would typically be required in p5.js. The `.inside` operator can be used to add

DOI: 10.2312/eurp.20171188

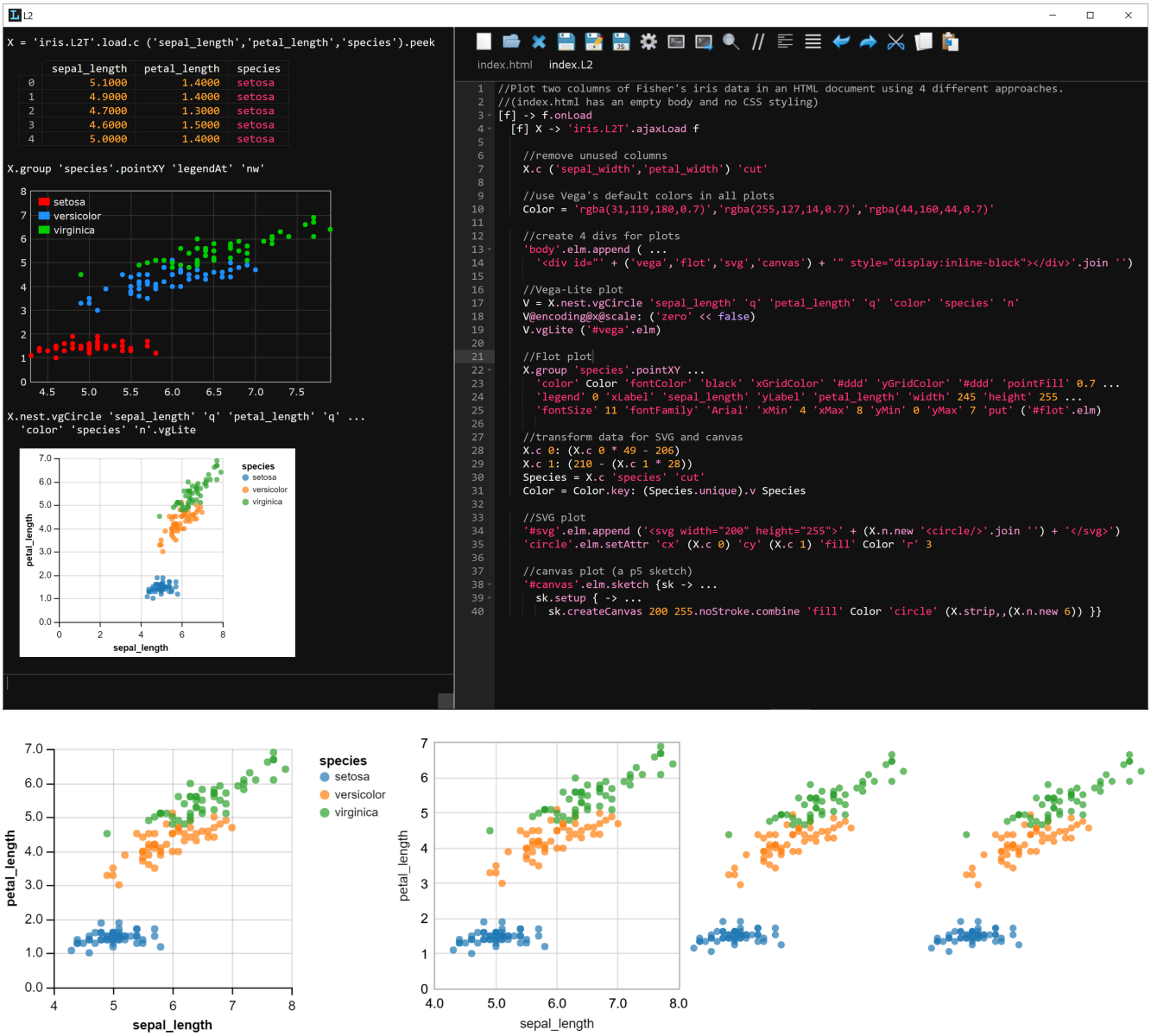

Figure 1: *Top: console and editor in the L2 Environment. Bottom: The HTML document corresponding to the file shown in the editor.*

event listeners to a variety of shape types including polygons and polylines.

3. Example

Figure 1 (top-left) demonstrates the use of Flot and Vega in the interactive console. The code in the top-right of Figure 1 generates HTML to display the same data set in the same style using all four of the approaches discussed above. The results are shown in the bottom figure. To keep the example brief, we have not included axes with the SVG and canvas plots, but the example is sufficient to demonstrate how easy it is to use SVG and the canvas.

4. Future Work

L2 is available at https://github.com/gjmcn/L2; it is already an effective tool for data wrangling, basic data analysis, visualisation and front-end development. Future work will focus on improving the L2 Environment, adding to the core language where appropriate and writing high quality libraries—for machine learning and statistics in particular. We will also work to raise awareness of the project and to grow an active and engaged user base. Naturally, we hope that users will be keen to contribute libraries of their own and in some cases, contribute to the development of L2 itself.

References

[Lau] LAURSEN O.: Flot. http://www.flotcharts.org/. Accessed: April 9, 2017. 1

[McC] MCCARTHY L.: p5.js. https://p5js.org/. Accessed: April 9, 2017. 1

[nwj] Nw.js. https://nwjs.io/. Accessed: April 9, 2017. 1

[SMWH17] SATYANARAYAN A., MORITZ D., WONGSUPHASAWAT K., HEER J.: Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 341–350. 1

[SWH14] SATYANARAYAN A., WONGSUPHASAWAT K., HEER J.: Declarative interaction design for data visualization. In *Proceedings of the 27th annual ACM symposium on User interface software and technology* (2014), ACM, pp. 669–678. 1