Journal of
open research software

Bryant, N and Wildfire, J 2016 Webcharts – A Web-based Charting Library for Custom Interactive Data Visualization. *Journal of Open Research Software,* 4: e29, DOI: http://dx.doi.org/10.5334/jors.127

SOFTWARE METAPAPER

Webcharts – A Web-based Charting Library for Custom Interactive Data Visualization

Nathan Bryant[1] and Jeremy Wildfire[2]

[1] Lead Designer and Programmer, Rho, US

[2] Project Lead, Rho, US

Corresponding author: Jeremy Wildfire (jeremy_wildfire@rhoworld.com)

Webcharts is a JavaScript library built on top of D3.js that creates reusable, flexible, interactive charts that are highly customizable. Webcharts provides a method for creating commonly-used charts, including bar charts, scatterplots, and timelines, through a simple configuration scheme. Charts created with Webcharts allow users to dynamically manipulate chart data, appearance, and behavior both through callback functions and input elements that are tied to chart objects. This approach allows users to create reusable charts that range from simple static graphics to complex interactive data exploration tools with custom user interfaces, all using the same library.

Keywords: Charting library; Data visualization; D3.js; JavaScript
Funding statement: This work was funded in part by NIH grant UM2AI117870.

(1) Overview

Introduction

Background

Statistical graphics have been in use since at least the 1600s [1, 2], but work to understand the links between visualization and human perception, and to create visualization "best practices", is a much more recent development [2–5]. This research, combined with insights into human-computer interaction [6] and the exponential growth of computing power, has laid the foundation for the release of open-source software packages for data visualization in the late 2000s [7–9]. Released in 2011, the Data-Driven Documents JavaScript library, or D3.js, provides a process for manipulating web documents based on data, which allows users to create interactive data visualizations on the web [10]. D3.js has since been documented [11] and has attained widespread use [12]. At its core, D3.js facilitates the manipulation of the Document Object Model based on data. No full-fledged charting functions are provided in the core D3, but a suite of functions for creating scales, manipulating data and creating axes are included for easy chart creation; this functionality has led to the creation of hundreds of examples and over a dozen charting libraries built using D3.js [13].

Many charting libraries have been written based on D3.js. Most of these focus on providing custom wrapper functions for generating charts with just a few lines of code. These libraries range from functions used to create specified chart types to extremely broad, declarative visualization grammars which provide a standard process for creating a wide variety of custom visualizations [14–16]. Many existing libraries have prescribed options for basic interactivity (filtering, zooming, etc.) and customizations (changing colors, adding annotations, etc.). Libraries may allow users to write custom JavaScript around a given chart, but adding interactivity outside of the intended use of a library is often impossible without editing the source code or reverse engineering the visualization using the base D3.js library.

Approach

Webcharts combines a fairly broad charting framework with a functional, event-driven approach to interactivity. Three central objects are combined with the concept of a 5-step chart lifecycle to create an environment where data visualizations can be created using initial, declarative configurations and then extended with custom functions. This enables a wide range of customization within the Webcharts framework, which allows for complex, interactive charts. Further, care has been taken to reduce dependencies and ensure that the Webcharts library is compatible with both modern web browsers and the Node.js environment.
The three objects in Webcharts are:

1. Chart – A chart with conventional x- and y-axes, rendered with SVG. Each chart is created by calling a function using the following parameters:
   a. A CSS selector that identifies the DOM element in which to create the chart (required)
   b. A configuration object whose properties describe the chart's behavior and appearance (required)

   c. A controls object, described below (optional)

   The chart is then passed a dataset upon initialization.
2. Table – A simple table, rendered as an HTML <table> element with a <thead> and <tbody>. A table is created by a function using the same parameters as above.
3. Controls – A set of inputs, rendered with <input> and <select> elements. Controls manipulate charts and/or tables by changing its configuration (thus changing its appearance or behavior) or by filtering the underlying data.

The Webcharts objects are rendered via a standard 5-step process that includes steps for the Creation, Initialization, Layout, Drawing and Resizing of the objects. Each step in this life cycle can be refined using insertion points for custom code, implemented as callback functions. A set of Chart, Table, and/or Controls objects can be packaged together with a collection of such callbacks to create powerful templating functions. This approach allows for easy reproduction of highly-complex visualizations, as demonstrated by the examples shown below.

Implementation and architecture

Loading Webcharts

Webcharts can be used in modern browsers (Chrome, Firefox, IE9+, etc.) and also exports itself as a CommonJS module for compatibility with Node. Install the package via npm:

```
npm install --save webcharts
```

Then, use it in your modules:

```
var webCharts = require('webcharts');
// or, in ES6:
import webCharts from 'webcharts';
```

To use Webcharts in the browser, just make sure to include a reference to D3 first:

```
<script type='text/javascript'
src='http://d3js.org/d3.v3.min.js'></script>
<script type='text/javascript'
src='webcharts.js'>
</script>
```

Webcharts can also be used with an AMD module loader like Require.js:

```
require.config({
  paths: {
  webCharts: 'webcharts'
  }
});
require(['webCharts'], function(webCharts) {
  console.log(webCharts.version);
  // make some charts!
});
```

Making a Chart

Once Webcharts and D3.js are loaded, a chart is created with a call to `webCharts.createChart()`. The function returns an object that represents a chart. The code to initialize a simple scatter plot is given below. See http://bl.ocks.org/nbryant/aeaf8d734d7600ca3afa for a live example:

```
var settings ={
  "max_width":"500",
  "x":{
    "label":"Protein (g)",
    "type":"linear",
    "column":"Protein (g)"
  },
  "y":{
    "label":"Carbs (g)",
    "type":"linear",
    "column":"Carbo(g)"
  },
  "marks":[
   {
    "type":"circle",
    "per":["Food"],
    "tooltip":"[Food]"
   }
  ],
  "aspect":"1",
  "gridlines":"xy"
};
d3.csv('calories.csv',function(error,csv){
  webCharts.createChart('body', settings).
  init(csv);
})
```

The first argument, "body", tells the function where to draw the chart. This is a simple CSS selector, so it may reference a DOM element name (like in this example) or class attribute, like ".chart-wrapper".

The second argument is a JavaScript object that sets a number of options for the chart. The config object in this example sets some basic options like: what dataset fields should be mapped to the x and y axes, what type of marks should be drawn, how wide the chart can get (max_width), its aspect ratio, and where gridlines should be drawn. All of the possible configuration options are described at https://github.com/RhoInc/Webcharts/wiki/Chart-Configuration.

The chart object returned by `webCharts.createChart()` can then be initialized by passing data to the chart via its `init()` method. The init method triggers the remainder of the chart's life-cycle (**Table 1**), including the Layout, Draw, and Resize phases.

The initial settings for a chart are established via a settings object during the Chart Creation step, and data is linked to the chart when the chart is initialized. However, both the settings and the underlying data can be modified throughout the remainder of a chart's lifecycle, either via a linked control object or custom callback functions. Changing a control object linked to a chart immediately updates the object by triggering steps 4 and 5 in the chart's lifecycle. More detail about the lifecycle of Webcharts objects can be found at: https://github.com/RhoInc/Webcharts/wiki/Webcharts-Life-Cycle.

Webcharts does not restrict the file size of the data loaded, but browser performance may be poor when

rendering large numbers of elements (issues typically start at around 10,000 elements). Standard techniques such as data aggregation and server-side rendering can be used in conjunction with Webcharts to improve performance with very large data sets.

Examples

While it can be used in any domain, Webcharts was principally designed to be used in clinical trial research, which is reflected in the examples below **Figures 1, 2** and **3**. Additional examples are available at https://github.com/RhoInc/Webcharts/wiki/Examples.

Order	Phase	Description
1	Chart Creation	Chart Object is created and returned via the `createChart()` method.
2	Initialization	The `init()` methods establishes default settings, binds data to the chart, and triggers the remaining steps in the life-cycle.
3	Layout	An svg is added to the DOM along with placeholder chart components (e.g. axes).
4	Draw	Raw data is processed as needed for charting. Pre-processing steps are completed (e.g. Scales are calculated.)
5	Resize	All chart marks are rendered/ updated.

Table 1: Chart Foundry Lifecycle.

Quality control

Webcharts has been tested in modern web browsers, including: Chrome, Firefox, Safari and Internet Explorer (version 9 and later). Full documentation and working examples are available at: https://github.com/RhoInc/Webcharts/wiki/. Additionally, outstanding issues, planned features, and support requests are tracked at: https://github.com/RhoInc/Webcharts/issues.

(2) Availability

Operating system

Webcharts does not target specific operating systems; it is compatible with modern web browsers (IE 9+, Google Chrome, Firefox, Safari, etc.)

Programming language

JavaScript

Additional system requirements

None

Dependencies

Data Driven Documents (D3.js) version 3

Figure 1: Simple Dashboard with Controls – This study monitoring dashboard shows common study metrics such as enrollment status, visit completion, and specimen and case report form status. The dashboard combines 6 different chart objects, several of which have attached controls, allowing users to filter by site and change the y-axis from a relative value to an absolute count. A simple callback function is used in the top left chart to customize the legend with overall counts and provide details for the y-axis in a tooltip. An interactive version of this chart, along with the source code, is available at: http://bl.ocks.org/jwildfire/raw/80890e1ff7bdc7f43079/.

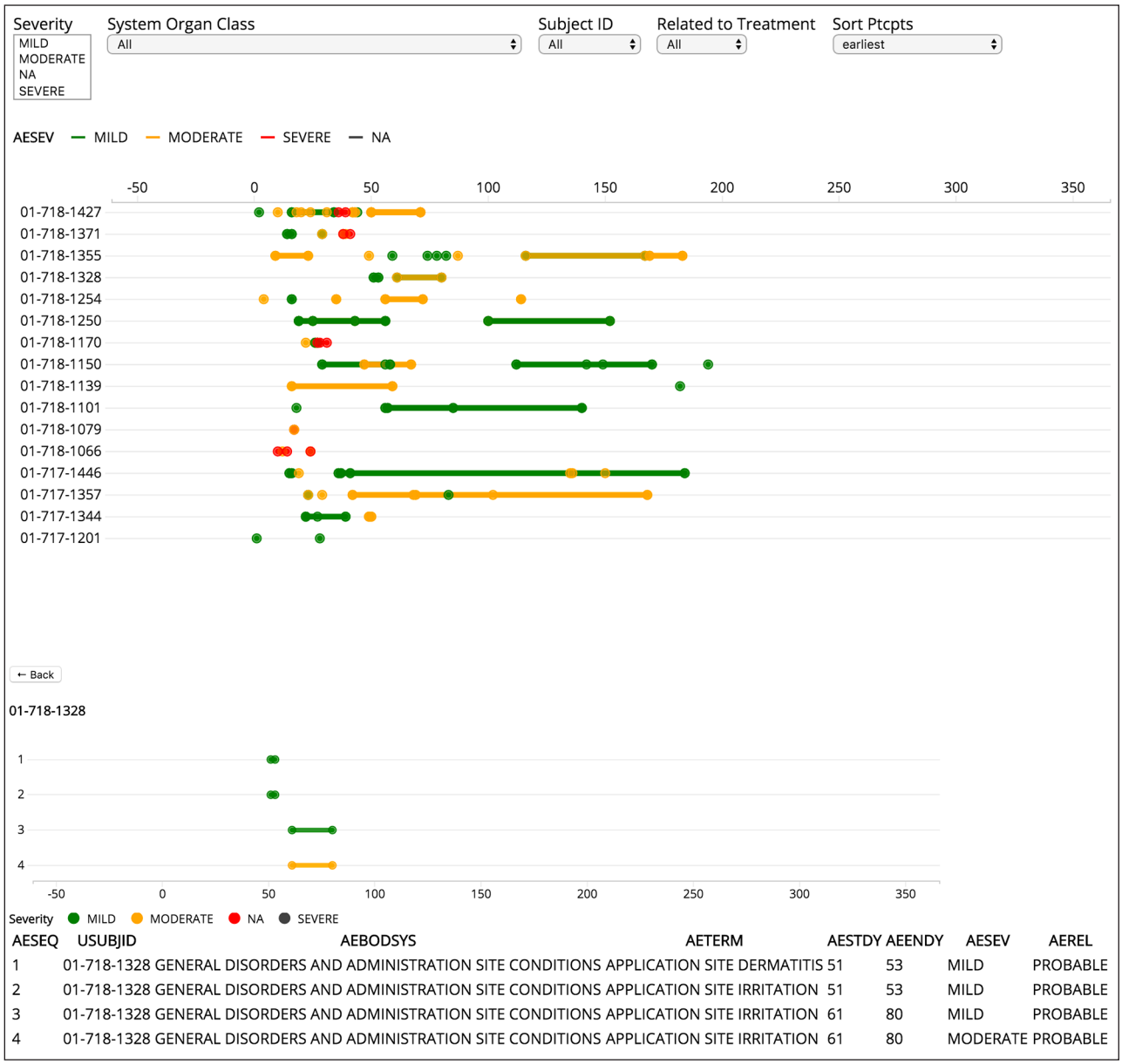

Figure 2: Panel A – Adverse Event Timeline – Overview. Panel B – Adverse Event Timeline – Participant Details – This figure combines 2 chart objects, a table object, and a control object, along with several custom callbacks to show the pattern of adverse events in a clinical trial. Panel A shows the pattern of adverse events over time for all participants. Panel B is triggered by clicking on a participant ID (shown as a label on the y-axis), and presents a detailed listing of adverse events for a single individual. While there are no dependencies other than Webcharts and d3.js, the code to produce this chart has also been saved as a separate library to facilitate re-usability. An interactive version of this chart along with the source code is available at: https://bl.ocks.org/jwildfire/raw/9865b8e7761f28a8a1557622d93ffbde/. The stand-alone Adverse Event Timeline library is at: https://github.com/RhoInc/ae-timelines.

List of contributors

Nathan Bryant, Lead Programmer and Designer
Jeremy Wildfire, Group Lead and Programmer
Ryan Bailey, Project Manager
Spencer Childress, Testing
Rich Budrevich, Testing
Britt Sikora, Coordination and Editing
All contributors are at Rho, Inc.

Software location

Archive

Name: Zenodo
Persistent identifier: 10.5281/zenodo.49396
Licence: MIT
Publisher: Nathan Bryant
Version published: 1.6.1
Date published: 5/6/2016

Code repository

Name: Github
Identifier: https://github.com/RhoInc/Webcharts
Licence: MIT
Date published: 8/31/2015

Language

English

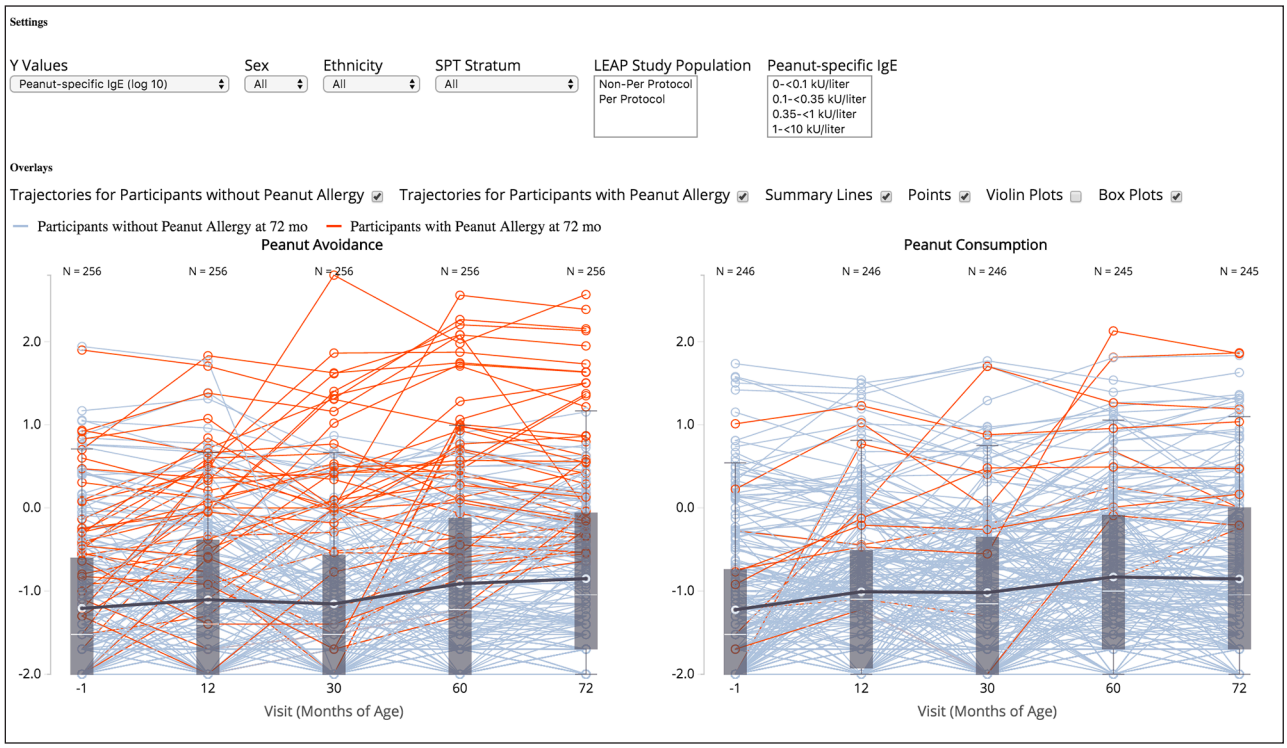

Figure 3: Immunologic Outcome explorer – This display is adapted from Figure 3 in the New England Journal of Medicine article, Randomized Trial of Peanut Consumption in Infants at Risk for Peanut Allergy [17]. The chart was originally created in response to reader correspondence [18], and was later updated to include follow up data in conjunction with a second article, Effect of Avoidance on Peanut Allergy after Early Peanut Consumption [19]. The interactive version allows the user to select from 10 outcomes on the y-axis. Selections for sex, ethnicity, study population, skin prick test stratum, and peanut specific IgE at 60 and 72 months of age can be interactively chosen to filter the data and display subgroups of interest. Figure options (e.g. summary lines, box and violin plots) can be selected under the Overlays heading to alter the properties of the figure. An interactive version of this chart is available at: http://bl.ocks.org/jwildfire/raw/893681c8f49470ee027a/.

(3) Reuse potential

Webcharts is open source and has a permissive MIT License that allows users to share and adapt the library. Although Webcharts was designed with a focus on clinical trial application, it can be used in any data domain.

Technically, the Webcharts library was designed to balance reusability and customizations. The code used to create a chart for a given dataset can often be applied to many other, similar datasets. If the structure of the input dataset is consistent, a chart can be rendered again and again with new data. The library has a single well-supported dependency (D3.js) and is compatible both with modern web browsers and the Node.js environment.

Acknowledgements

We would like to thank Herman Mitchell, Agustin Calatroni, Rich Budrevich, and Russ Helms for their guidance and support.

Competing Interests

The authors declare that they have no competing interests.

References

1. **Friendly, M** 2005 'Milestones in the history of data visualization: A case study in statistical historiography', in Weihs, C. and Gaul, W. (ed.) *Classification—the Ubiquitous Challenge*. Heidelberg: Springer Berlin Heidelberg, pp. 34–52. DOI: http://dx.doi.org/10.1007/3-540-28084-7_4
2. **Tufte, E R** 1990 *Envisioning information*. United States: Graphics Press USA.
3. **Few, S** 2004 *Show me the numbers: Designing tables and graphs to enlighten*. Oakland, CA: Analytics Press.
4. **Bertin, J** 1983 *Semiology of graphics: diagrams, networks, maps*. Madison, WI: University of Wisconsin Press.
5. **Cleveland, W S** 1993 *Visualizing data*. Hobart Press.
6. **Shneiderman, B** 1996 'The eyes have it: A task by data type taxonomy for information visualizations.' In *Visual Languages, 1996. Proceedings., IEEE Symposium on* 1996 Sep 3, pp. 336–343. DOI: http://dx.doi.org/10.1109/vl.1996.545307
7. **Wickham, H** 2009 *ggplot2: elegant graphics for data analysis*. Springer Science & Business Media.
8. **Sarkar, D** 2008 *Lattice: multivariate data visualization with R*. Springer Science & Business Media. DOI: http://dx.doi.org/10.1007/978-0-387-75969-2
9. **Bostock, M** and **Heer, J** 2009 'Protovis: A graphical toolkit for visualization.' *Visualization and Computer Graphics, IEEE Transactions on* 2009 Jun 15, pp. 1121–1128.

10. **Bostock, M, Ogievetsky, V** and **Heer, J** 2011 D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12), 2301–2309. DOI: http://dx.doi.org/10.1109/TVCG.2011.185
11. **Murray, S** 2013 *Interactive Data Visualization for the Web*. Sebastopol, CA: O'Reilly Media.
12. **Most-starred projects on Github [search results].** GitHub Website., https://github.com/search?q=stars%3a%3E1&s=stars&type=Repositories. Accessed March 25, 2016.
13. **Gallery · mbostock/d3 Wiki · GitHub.** Website., https://github.com/mbostock/d3/wiki/Gallery. Accessed March 25, 2016.
14. **Vega: A Visualization Grammar.** Website., https://vega.github.io/. Accessed March 25, 2016.
15. **Vega and D3.** Website., https://github.com/vega/vega/wiki/Vega-and-D3. Accessed March 25, 2016.
16. **ggvis 0.4 Overview.** Website., http://ggvis.rstudio.com/. Accessed April 9, 2016.
17. **Du Toit, G, Roberts, G, Sayre, P H, Bahnson, H T, Radulovic, S, Santos, AF, Brough, H A, Phippard, D, Basting, M, Feeney, M** and **Turcanu, V** 2015 'Randomized trial of peanut consumption in infants at risk for peanut allergy.' *New England Journal of Medicine*, *372*(9): 803–813. DOI: http://dx.doi.org/10.1056/NEJMoa1414850
18. **Noonan, L, Alpan, O, Alpan, OO, Du Toit, G, Roberts, G** and **Sayre, PH** 2015 'Peanut consumption in infants at risk for peanut allergy.' *The New England journal of medicine*, *372*(22), pp. 2163–2163. DOI: http://dx.doi.org/10.1056/NEJMc1504021
19. **Du Toit, G, Sayre, P H, Roberts, G, Sever, M L, Lawson, K, Bahnson, H T, Brough, H A, Santos, A F, Harris, K M, Radulovic, S** and **Basting, M** 2016 'Effect of avoidance on peanut allergy after early peanut consumption.' *New England Journal of Medicine*, *374*(15), pp. 1435–1443. DOI: http://dx.doi.org/10.1056/NEJMoa1514209

How to cite this article: Bryant, N and Wildfire, J 2016 Webcharts – A Web-based Charting Library for Custom Interactive Data Visualization. *Journal of Open Research Software,* 4: e29, DOI: http://dx.doi.org/10.5334/jors.127

Submitted: 22 April 2016 **Accepted:** 01 July 2016 **Published:** 19 July 2016

]u[*Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press

OPEN ACCESS