2015

Evaluation and Integration of a JavaScript Graphing Library for a Nationwide Honey Bee Hive Scale Network

Emily Johnson
Grand Valley State University

Follow this and additional works at: http://scholarworks.gvsu.edu/cistechlib

Recommended Citation
http://scholarworks.gvsu.edu/cistechlib/208

This Project is brought to you for free and open access by the School of Computing and Information Systems at ScholarWorks@GVSU. It has been accepted for inclusion in Technical Library by an authorized administrator of ScholarWorks@GVSU. For more information, please contact scholarworks@gvsu.edu.
Evaluation and Integration of a JavaScript Graphing Library for a Nationwide Honey Bee Hive Scale Network

By
Emily S. Johnson
April 23, 2015
Evaluation and Integration of a JavaScript Graphing Library for a Nationwide Honey Bee Hive Scale Network

By
Emily S. Johnson

A project submitted in partial fulfillment of the requirements for the degree of
Master of Science in
Computer Information Systems

at
Grand Valley State University
April 23, 2015

Dr. Jonathan Engelsma
April 23, 2015
Table of Contents

Abstract ..............................................................................................................................4
Introduction .....................................................................................................................4
Background and Related Work ......................................................................................5
Program Requirements .................................................................................................6
Implementation ...............................................................................................................8
Results, Evaluation, and Reflection .............................................................................11
Conclusions and Future Work ......................................................................................12
Bibliography ..................................................................................................................14
Appendices .....................................................................................................................15
  Appendix A: Scoring Rubric and Results .................................................................15
  Appendix B: Three-Series Chart with Legend & Navigator .....................................16
  Appendix C: Three-Series Chart with Annotations & Navigator ..........................17
  Appendix D: Chart showing invalid zero-weight points .........................................18

List of Figures

Figure 1: Vertical Zoom in Dygraphs ............................................................................9
Figure 2: Custom Series Toggle in Dygraphs ..............................................................9
Figure 3: Annotation Popover in Dygraphs .................................................................9
Figure 4: Annotation Range in Dygraphs ....................................................................9
Figure 5: XY Zoom in Highcharts ..............................................................................10
Figure 6: Legend & Series Toggle in Highcharts .........................................................10
Figure 7: Annotation Details in Highcharts ...............................................................10
Figure 8: Cropped Graph for PDF in Highcharts ......................................................11
Abstract
Since 2006, an unprecedented and alarming decline of honey bee populations has puzzled the apiculture community: approximately 30% of the bee population has been lost each winter. Since upwards of one third of the nation’s food is directly or indirectly produced by bees, this rate of depletion is cause for concern. The goal of the Bee Informed Partnership is to better evaluate, understand and reduce bee colony losses. One of their programs centers around a Hive Scale Portal being co-developed with GVSU’s School of Computing and Information Systems. This network is designed to monitor key hive metrics and collect this data nationwide for research purposes. A web portal that gives beekeepers and entomologists insight into their data accompanies the hive scales deployed in the field. A key part of the Hive Scale Portal is an interactive graph displaying hive data collected 24x7 from live colonies in the field. However, the original graphing software (Dygraphs) used by the portal was limited and in need of visual and technical improvements. The purpose of this project is to enhance the portal by evaluating a series of JavaScript graphing libraries, selecting/implementing the best candidate, and extending or customizing its functionality as needed. A total of ten different open-source graphing packages were evaluated based on the project requirements, and Highcharts selected and integrated in the end.

Introduction
Since 2006, an unprecedented and alarming decline of honey bee populations has puzzled the apiculture community: approximately 30% of the bee population has been lost each winter. Since about one in three bites of food we eat is directly or indirectly pollinated by honey bees, this rate of depletion is cause for concern. In order to spread awareness of the issue, collect research data, and search for a solution, the Bee Informed Project (BIP) was initiated by Dennis vanEngelsdorp at the University of Maryland in 2011. The project is funded by the United States Department of Agriculture and the National Institute of Food and Agriculture, and also involves Penn State University, Appalachian State University, Grand Valley State University, and several others. It is an ongoing effort that will expand NASA’s original program, HoneyBeeNet, which proposed a nationwide network to monitor nectar flow and plant phenology along with climate. “The Bee Informed Partnership is dedicated to working with beekeepers to better understand which management practices work best,” the website says. BIP uses an approach similar to large-scale cancer research done in the medical field: “We gather survey data from thousands of beekeepers every season to understand how different management practices affect honey bee health, and we report our findings back to the industry.”

Anne Marie Fauvel, a Liberal Studies professor and biologist at Grand Valley State University, took an interest in the issue and sought to find a way to contribute. “During the winter 2012 semester, before our bees arrived, I enlisted the help of a team of senior engineering students to develop and build a digital, solar-powered scale,” she says, “and a second team of senior computer and information systems students to develop the software to automatically collect and disseminate the data to a website built around the project.” The software team was advised by a fellow bee enthusiast and CIS professor Jonathan
Engelsma. This hardware and software prototype became the start of GVSU’s involvement with the Bee Informed Partnership. In order to provide a market ready-device with a high enough production capacity, BIP identified commercial vendors to manufacture the hardware using the Application Programming Interface (API) developed by GVSU. To encourage innovation and maintain fair prices, BIP didn’t limit the scale to just one vendor. The initial vendors included SolutionBee, who built their product according to BIP’s specification, and BeeWatch, out of Denmark.

Any beekeeper can purchase and deploy one of these scales, register with BIP, and access his or her live hive data through a Ruby On Rails web application called the Hive Scale Portal. They also have the option to make their hive scale data public and visible to users of the BIP site. “Being part of this project makes the data doubly productive as researchers and scientists who tap into this data stream may learn more about the linkages between nectar flows, nosema disease, varroa populations and other colony health issues” says Fauvel. A key part of the Hive Scale Portal is an interactive graph of data from scales deployed in the field. Up to this point, graphing in the portal was done using Dygraphs, an open source JavaScript library created in 2011 by a Google developer. While Dygraphs fulfilled most of the portal’s requirements, it had some technical limitations which motivated the need for an alternative. The purpose of this project is to enhance the portal by evaluating a series of JavaScript graphing libraries, implementing the best candidate, and extending or customizing its functionality as needed.

Background and Related Work

The BIP Hive Scale Portal site was built in Ruby on Rails and is hosted by Heroku, a third party hosting service. The site was already fully functional at the outset of this project – users could create an account, register a scale, and interact with the data using the Dygraphs graph. The scales record temperature, weight, and humidity as often as every 15 minutes. Beekeepers can create annotations on the graphs to denote external events impacting the hive data; for example, a decrease in weight due to harvesting honey or splitting a colony, or an increase in weight due to feeding or precipitation. Entomologists can also access the data and use these annotations to understand anomalies in the data. Since some of these anomalies are gradual (such as falling or melting snow), one annotation may not be enough to reflect the change. The BIP team requested a way to annotate a range of data points for situations like this.

The scale hardware itself is also subject to external factors, mainly losses of data communication or power. The current scales on the market depend on either Bluetooth or wide-area cellular links to send information, both of which have disadvantages. Bluetooth requires the beekeeper to physically come within 30 feet of the hive to capture data, so it is not a fully automated process. With cellular data, the beekeeper is required to subscribe to a data plan from a carrier with coverage in the area. As Anne Marie Fauvel experienced, forgetting to pay the bill can result in a loss of data. Honey bee hives are often located in fields and far from buildings, so the scales are battery powered, which means the batteries drain and eventually die. Some batteries are not durable enough for Midwestern/northern climates, and can
malfunction (inaccurately recording “0” for weight) when the temperature gets too cold. In a typical Michigan February, this can happen often and causes the weight on the graph to appear to be constantly spiking and plummeting. The BIP team requested a way to allow these erroneous points to be hidden from the graph.

There were other suggested enhancements that the Dygraphs implementation did not include, such as a way to export a graph image or generate a printable report. These limitations were used to create a rubric for evaluating other JavaScript graphing libraries and identifying the one, Dygraphs or otherwise, that was best suited for the portal site.

**Program Requirements**

The first phase of the project was to identify and evaluate various graphing libraries. Benjamin Dummer, a collaborator from Appalachian State University, has worked on graphs in other parts of the BIP site and explained that “it really depends on what type of data you are trying to visualize, how much interactivity you desire, how quickly you want to get up and running vs. extensibility, and other factors.” He also advised to keep browser support, mobile-friendliness, loading times, library size, community size, and long-term stability in mind. In their selection process, Ben and his colleagues were looking for “technology that would continue to grow and wouldn't go stale quickly.” Following Ben’s recommendations and the Hive Scale Portal requirements, the following graphing library criteria were defined and weighted from one to five (low to high). (See Appendix A for results.)

1. **Visualization**
   a. Horizontal and Vertical Zoom (5). The existing graph implementation allowed for a linear time range to be selected and enlarged. Due to the data series having large vertical gaps between them, the horizontal zoom did not allow for significantly greater detail to be seen.
   b. Toggle of Series Display (4). Similar to the lack of vertical zoom, visual detail was limited due to the permanent display of all series on the chart – Temperature, Weight, and Humidity.
   c. Toggle of Annotation Display (4). The existing implementation plotted user-defined flags, or annotations, directly on the chart. The flags were numbered so that a user could manually find the corresponding detail row in a table beneath the graph. There was no option to hide the flags on the graph or interact with them in any way.

2. **Manipulation and Annotation**
   a. Annotations (5). Beekeepers were given the ability to select a specific data point on the chart and record additional detail, both for their own documentation and for communication with entomologists who would later access the data. These annotations typically include details of equipment changes, feeding of the colony, or freeform text to explain data anomalies due to precipitation or other external influences.
b. Intuitive Interface (5). Users of the BIP Hive Scale Portal vary in level of technical proficiency, so a highly usable interface was desired.

c. Ability to annotate a range of data points (5). Certain instances which require annotation cannot be adequately defined by a single data point, such as a period of heavy snowfall. The existing implementation did not allow for a range of points to be annotated.

d. Delete or Invalidate Ranges or Sets (4). Some periods of time can be considered invalid, such as a span of time where a hive has not yet been fully assembled. Previously, users could select a time range and permanently delete its data.

3. Compatibility

a. Compatibility with the BIP Hive Scale Portal site (5). The Hive Scale Portal site was already implemented and hosted in a production environment. The site was built in Ruby on Rails and hosted by Heroku, a third party service. A new graphing library would need to be gracefully integrated into the site without impacting the existing functionality.

b. Customizable Style (4). Along with technical compatibility, the new graphs would also need to be visually consistent with the rest of the site.

c. Multi-Browser Friendly (4). Users access the site with their own personal or institutional devices, which can vary in technical capability. The new charts would need to be rendered on various browsers, operating systems, and screen sizes.

4. Bonus

The following features were not required, but would add to the convenience and overall evaluation of a library:

a. Direct Print Option (4). The ability to generate a polished report on the data would be of use to end users, particularly the entomologists.

b. Mobile Friendly (4). Apiaries are generally outdoors and often in rural locations. Users may not always be accessing the data with a traditional desktop or laptop computer, so a responsive mobile view of the graph was desired.

c. Ruby Gem (1). Ruby features the concept of “gems”, or prepackaged web application add-ons. Some libraries offer gems as an installation method. Gems make it straightforward for a developer to incorporate these add-ons by simply adding a line to the Rails application’s gemfile.

d. Extensibility (2). Some JavaScript/other libraries are designed with future developers in mind, and are written in a way that allows others to customize and expand upon their code. Because it may be difficult to find one library that met all of the criteria, an easily extensible library was ideal.

e. Documentation (4). Libraries with thorough documentation can be implemented faster and with fewer errors. This also aids in the transition of code management to other developers, especially in a large project.
f. User Community Size and Activeness (2). A larger user base generally leads to higher quality and more frequent updates, and also provides a resource for peer technical support.

Since the BIP Hive Scale Portal already had working graphs, the site’s existing library (Dygraphs) would be evaluated alongside others to potentially reduce the risk and impact of implementing an entirely new library. Although Dygraphs did not have all the desired functionality, the developer had released updates since it was first installed and there was potential to extend it to fulfill the site’s unmet needs.

**Implementation**

There are many JavaScript graphing libraries available – some are modern and actively being developed, while some are technically outdated or have been abandoned. Benjamin explained that “Most of the libraries fall into one of three groups: those that depend on jQuery, those that depend on D3.js, and those that are self-contained” \(^\text{11}\). He noted that their team had selected D3 and a small library called Dimple.js, which they mainly use for pie and bar charts, and that the Portal’s time-series plotting requirements may result in a different selection.

D3, or Data-Driven Documents, is “a JavaScript library for manipulating documents based on data”, including HTML, CSS, and SVG elements \(^\text{12}\). D3 is extremely powerful and can create infinite types of visually striking charts and graphs. It is similar to jQuery in its element selection and document modification, but it’s more geared toward generating SVG code and visual effects. D3 is more of a low level language than a traditional graphing tool, so libraries that depend on D3 were evaluated more closely than D3 itself. These candidates included Dimple.js \(^\text{13}\), C3.js \(^\text{14}\), and Rickshaw \(^\text{15}\). Dimple’s main downfall in the BIP case was that it doesn’t have any kind of annotation or flag that can be displayed on a chart. C3 had the same issue. A user forum \(^\text{14b}\) discussed the potential to add annotations, but it would be completely custom and created entirely by the developer. Rickshaw’s visual style was polished and nice-looking, but it appeared to have been abandoned by its creator \(^\text{15b}\).

The other libraries that were explored are Highcharts \(^\text{16}\), vis.js \(^\text{17}\), and Dygraphs (the “incumbent”). Highcharts is a very professional and complete library, and has excellent documentation with frequent updates. As Highcharts was the “winner”, it will be discussed in more depth later. Vis is fairly well documented, but there wasn’t much active development or activity from its user base.

Since the existing code with Dygraphs was fully functional, its additional capabilities were explored before testing any other frameworks. Dygraphs had made some enhancements since it was originally implemented, and there were also opportunities to extend its out-of-the-box functionality. All of the model and controller code could be repurposed, as well as most of the JavaScript. The key features that Dygraphs was lacking were vertical zoom, toggling series display, hiding “zero weight” points, showing annotation details, annotating date ranges, and exporting a graph image. It turned out that Dygraphs did have a vertical zoom option, but it was incompatible with another option (a sliding time range selector...
along the X-axis) the existing graph had configured. By disabling the time range selector, Y-axis (vertical) zoom was available (Fig.1).

The ability to toggle a series on or off, for example clicking “Temperature” on a legend to hide that specific plot line, was not an available option in Dygraphs. This functionality was added by creating a set of checkboxes, each corresponding to a series on the graph, along with a JavaScript function that hid or showed the lines when checked (Fig.2).

Adding the “hide zero points” functionality was successful as well. This was done by adding a button that sets a Boolean value, retrieves graph data again, and excludes zeros in the results if true. The Bootstrap’s features is called a popover, and it “pops over” a small dialog for detail when clicked. This was useful for adding detail to annotations in Dygraphs. Dygraphs does not allow adding actions to flags on the chart, but it does allow CSS classes. After drawing the initial chart, jQuery was used to locate all of the points with a certain CSS class, and add a Bootstrap popover to them with more detail (Fig.3).

For the database to support annotation ranges, David Qorashi, the developer responsible for the site’s back-end, added an “xval2” column to the Annotation table. Single-point annotations would just use the standard “xval”, while ranges would use both columns. The Rails controller was then updated to accept and use the xval2 value. On the front-end, the functionality was similar to the “hide zeros” button. A new “Annotate Range” button was added, which set a Boolean. The script didn’t take any further action until the user highlighted a range (in the same way they would zoom in on the graph). The script would then intercept the default zoom action, take its minimum and maximum x-values to use for the range, and prompt the user for annotation details instead (Fig.4).
The last missing piece of the Dygraphs implementation was the option to export a graph image to a file or print some kind of report. However, it does have an “Export.AsCanvas” function, which creates an HTML5 canvas out of the graph. This option was utilized in a custom function to generate a printable report. After drawing an image on a canvas, the canvas could then be translated into a Data URI and then into a standard image. A library called jsPDF was then used to render the image, along with header information, onto a new PDF document. jsPDF has an add-on module for using PNG images, so that was included as well. The concept of jsPDF is extremely simple, however using it was rather painstaking, as every bit of text must be plotted in relation to the top left corner of the document.

Highcharts is a family of JS graphing libraries, also including Highstock (designed for Stock Market data), and Highmaps. Highstock was one of the few libraries evaluated that natively supported annotations, or “flags”, which was an important requirement for BIP. It did not include any support for adding flags on the fly, but its rich API had extensive documentation available. It has an “exporting” module, which includes preset buttons to export a graph to a JPG image, SVG, or a PDF. Highcharts is also the only candidate that required licenses to be purchased. Fortunately, it is available at no cost for education and to nonprofit organizations (including BIP). Note: Technically the library used was Highstock.js, but it will be referred to as Highcharts because many of the configuration options overlap.

All of the custom functions that were added to Dygraphs were adapted to also work with Highcharts. X-Y zoom is a convenient option available by default: instead of selecting only horizontally or vertically, the user can draw a rectangle on the exact part of the chart to which he wants to zoom (Fig.5). Series visibility toggle is another feature available by default. The user can click an item in the legend to show or hide that series on the chart (Fig.6 & Appendix B). Additionally, hovering over an annotation on the chart displays its details, without the need for any custom code (Fig.7 & Appendix C).

One powerful aspect of Highcharts is that it allows for customization of most chart events in a very transparent and well-documented way. Range annotations were implemented by customizing the “selection” event. The default selection behavior is to zoom in, but instead, the BIP graph can capture the start/end times and prompt the user to create an annotation. This is controlled in the same way that it was in the Dygraphs implementation – the user must first toggle the “Annotate Range” button to avoid inadvertently creating annotations. The “Hide Zeros” button is similar. In the first iteration of Dygraphs enhancements, clicking the button would result in the whole data set being processed again to weed out
zero weight points. In the Highcharts phase of the project, this was implemented a little differently. To preemptively prepare data for hiding zeros, the JavaScript simply creates one more array of data as an alternate source on the chart. When the button is toggled, the weight series data is substituted with the other set, which already has no zero points (Appendix D).

The last major Highcharts customization was the PDF export. Most of the groundwork had been done during the Dygraphs phase, so a lot of that code could be reused, however it was slightly simplified this time around. Instead of hard-coding every piece of data’s coordinates, variables were created to store top margin and left margin values, as well as offset distances for other pieces of data. This made it much easier to tweak the layout so that everything would fit nicely. Since Highcharts has a rich exporting module, it can natively be rendered to PDF, JPG, SVG, and PNG. The getSVG function was used to draw the graph onto an HTML canvas, similar to what was done in Dygraphs. To save space on the generated PDF document, some parts of the image were cropped out. This was achieved by drawing the SVG onto the canvas in two pieces: the chart and the legend. The title and buttons above the chart were left out, as well as the navigator between the chart and the legend (Fig.8).

Another improvement was made while Highcharts was being tested, which is to utilize Ajax to fetch a new date range for the graph without requiring the whole page to be reloaded. This is more convenient for the end user in multiple ways. There is less of a delay in retrieving new data sets, and preferences like “Hide Zero Points” can stay toggled after the new data set is loaded.

Results, Evaluation, and Reflection

Initially the task of finding and implementing a JavaScript graphing library seemed like it would be fairly simple and wouldn’t take much time. I quickly learned that there were many more options than I had estimated, and that there most likely wasn’t one that would fulfill all of the requirements. Another challenge was working with existing functioning code in Rails, where my skills were a little rusty. I had to study it and experiment with it quite a bit before being able to effectively alter it. This became easier over time as I learned what different parts of the code were responsible for and recognized what was safe to
change. Highcharts was a good outlet for experimenting with extending other people’s code. It ended up being a great learning experience, because the library really encourages customization, and I gradually became comfortable with making modifications to its default functionality. In the future, I’ll be less inclined to let the boundaries of someone else’s code become limitations in my projects.

When I had finished the implementation, I revisited some of the other candidates to collect more detail and begin to summarize the whole process. After having worked extensively with the Highcharts code and learning more about HTML5 canvases and SVG, some elements of the other libraries looked familiar and they appeared more feasible than they did at first. My usual learning method is to hastily skim through documentation and look for examples, but in this project I’ve made a habit of reading the documentation thoroughly for better understanding. I have also found github to be a valuable resource, and frequently read through the issues to gauge development activity or find solutions to problems.

There were a couple snags when integrating my updates into the site’s main code base. For the majority of my development, I had been working with a sample data set and didn’t have the most current records. When we merged the code and tested the charts on real data, some time periods were missing or showing in terms of milliseconds since 1970. The database timestamps were in UTC, but my localhost was in EDT and somewhere the code was overcompensating for the difference. There had been a time change since my last data export, which appeared to be skewing the data. David and I eventually resolved the issue by changing a Highcharts global option called UseUTC.

In April, the new Highcharts based implementation was successfully merged with the project’s master branch and tested via a pilot deployment on our staging server. After a small set of existing users confirmed the readiness of my implementation, we deployed to BIP’s production portal (http://hivescales.beeinformed.org). At this point, the new updated site is being used by beekeepers and bee scientists across the USA (as well as some limited usage in Europe!).

**Conclusions and Future Work**

So far, the users who have seen the new graphs have given positive feedback and like the new smoother look of the charts. The added functionality to annotate date ranges and hide erroneous zeros has improved the quality of graph data for end users, and the export functionality will allow for the data to be shared easily. There are 25-30 scales in the field today, but BIP hopes to increase this number to at least 100 scales in their network.

As for the honey bee population decline, current research is pointing to a complex combination of contributing factors. “It’s becoming more and more apparent,” Michigan beekeeper John Ebers states, “there’s a number of reasons, but of those, four or five chemicals seem to be the common denominator.” One class of pesticides called neonicotinoids, designed to ward off leaf-eating insects, has proven to have negative effects on honey bees as well as many types of birds and earthworms. These pesticides can harm a bee’s sense of navigation and ability to navigate back to its nest, as well as make it more susceptible to disease. Another factor is fungicides, which originally were thought to be harmless to bees. A recent study
showed that bees that ate pollen contaminated with these fungicides were in fact three times as likely to be infected by a parasite called Nosema ceranae\textsuperscript{19}.

The disappearance of honey bees is a complex problem that will require a complex solution. The Bee Informed Partnership conducts apiary health studies, longitudinal disease monitoring, and national management surveys to collect as much information as possible in pursuit of answers to this problem. With the knowledge they have accrued, BIP also offers educational resources to promote keeping healthier bees as well as several ways for beekeepers to participate in their research. “I think there’s a lot of promise here, but there’s also still a lot of work to be done if we’re going to solve the problems facing honey bees,” says Engelsma.\textsuperscript{7} With modern technology like hive scales and the massive amounts of data they can collect, scientists studying the honey bee population decline problem in the future will be able to apply big data and machine learning techniques to hopefully understand and begin to reverse the trend of colony loss.

Other Links of Interest:
## Appendices

### Appendix A: Scoring Rubric and Results

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Weight</th>
<th>D3</th>
<th>Dimple</th>
<th>C3</th>
<th>Rickshaw</th>
<th>Highcharts</th>
<th>Highstock</th>
<th>vis</th>
<th>Dygraphs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Visualization</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Horizontal/Vertical Zoom</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Toggle Series on-off</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Show/Hide Annotation Details</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td><strong>Manipulation/Annotation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Annotations</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Intuitive interface</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Annotate range of data points</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Delete or invalidate ranges or sets</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Compatibility</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compatibility with rest of site</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Multi-browser friendly</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Customizable Style</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td><strong>Bonus</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direct Print Option</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mobile friendly</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Ruby gem</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Extensible</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Good Documentation</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Community Size/Activeness</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>62</td>
<td>37</td>
<td>29</td>
<td>37</td>
<td>39</td>
<td>44</td>
<td>52</td>
<td>41</td>
<td>33</td>
</tr>
</tbody>
</table>
Appendix B: Three-Series Chart with Legend & Navigator

Dygraphs

Highcharts
Appendix C: Three-Series Chart with Annotations and Navigator

Dygraphs

Highcharts (with annotation details)
Appendix D: Chart showing invalid zero-weight points

Dygraphs

Highcharts:

Highcharts, with “hide zero” selected: