Google Summer of Code - Elastic
Kibana Project Proposal

Who you are?

Name: Hanqing Zhao

Country/Location: United States, Saint Paul, Minnesota

Field of Study: Computer Science master student in the University of Minnesota.

Experience: I have experience developing React and Node.js applications for the websites of TEDxKids@Xueyuanlu. I also develop my personal website in React.

Github Profile: https://github.com/aaronoah

Email Address: aaron.elite1993@gmail.com

Phone: +1 (612) 478-8386

Have you been applied to GSoC before?: No. First time GSoCer.

Which Project you want to work on?

Title: **Kibana: Calendar Visualization and Filtering**

Synopsis: The Kibana Visualize app now needs a new calendar visualization that is a specialized form of a heatmap, which provides single-day or multi-day events markings in varying colors based on different metrics. It should play well with the visualization editor, filtering and Dashboard.

Mentors: Tim Roes tim.roes@elastic.co, Thomas Neirynck thomas@elastic.co, Daniel Schneiter daniel.schneiter@elastic.co, Philipp Krenn philipp@elastic.co

Project Link on Github website: https://github.com/elastic/gsoc#kibana_calendar_visualization

How you will implement it?

Goals:

- Implement a new calendar visualization based on existing heatmap visualization along with its editor in the `Visualize` app with new EUI framework.
- Test the calendar visualization rendering and interactions with lots of data in terms of Editor changes, Filtering, various Dashboard functions.

Expected Results:

- A well-functioned calendar visualization with new EUI.
- Calendar visualization plays well with editor, filters, dashboard.

Timeline: (level of importance, the lowest number has the highest priority)

Time Frame	Start Date	End Date	Tasks	Importance
Community Bonding week 1	Apr 23rd		Bonding with mentors as well as the driver for the calendar visualization feature.	1
week 2			further requirements gathering and more Elasticseach internals learning.	1
week 3		May 14th	setup the additional environments and tools if needed for project preparation.	1
Coding week 4	May 14th	May 20th	● Follow the instructions: https://github.com/elastic/kibana/blob/master/docs/development/visualize/development-create-visualization.asciidoc Note: (to pass a React component in visConfig, should use component filed instead of template in the master branch codebase), Implement a new visualization type entry card (using Visualization Factory to initialize a config object with \`name\`, \`title\`, \`category\` (BASIC), \`image\`, \`stage\` (in lab or experimental), visConfig, editorConfig, etc). ● Register on core plugins - kbn_vislib_types (then it will be injected to the create visualization page), and design a \`Calendar\` visualization icon with Sketch. ● Test the new visualization existence in functional visualize app test in Mocha	2
week 5	May 21st	May 27th	● Follow the instructions: https://github.com/elastic/kibana/blob/master/docs/development/visualize/development-create-visualization.asciidoc , build a customized Visualization editor for calendar heatmap. by supplying \`collections\`, \`schemas\` and a new React EUI component for \`optionsTemplate\`.	1

			Note: (the editor default template views have too many components that can be reused for this visualization, it requires much more time to rewrite a new template for Calendar Visualization, so I will stick to the default template and use \`collections\`, \`schemas\`, \`optionsTemplate\`to finish this part.) ● The design of \`schemas\`: metrics can have the same aggFilters as heatmap and three buckets \`X-Axis\`: the beginning time to the end time in the format (Date Range or Date Histogram) and \`Y-Axis\`: the starting weekday to the end weekday (Monday - Sunday) (weekdays - defined). ● Run functional tests within the visualize app using Mocha, reuse some services like \`log\`, \`pageObjects\` to test aggFilters and Options.	
week 6	May 28th	June 3rd	● Implement a calendar chart class in two weeks: (**this feature is the core of the new calendar visualization**) ● Reuse visualization lib \`heatmap_chart\`, extends on it and override draw() with some data preprocessing (format to have weekdays, e.g. 2018-03-27 to Tuesday, Mar 27, 2018), import it in the series_types of point_series to provide a new render interface for the Handler. ● **Above thought details to been seen in Appendix** ● Test the various data for date that could cause formatting failure for next phase.	1
week 7	June 4th	June 10th	● Add addSquares() to paint the data cells for each date (including ignored cells). ● Add boundary stroke painting (the boundary between months) to the visualization lib to complete the draw() method.	1
Evaulations	June 11th	June 17th	● Write evaluations for mentors	2

week 8			● If time permitted, implement a similar visualize-legend React EUI componet that renders the legend color maps.	
week 9	June 18th	June 24th	● Implement the rest of visualize-legend React EUI componet that renders the legend color maps. ● Build the React component of visualization that takes care of obtaining vislib from registry and bind \`brush\`, \`click\`, \`renderComplete\` functions as well as rendering the \`visualize-legend\`, use chart_title to draw on each chart upfront with the calendar year (like Appendix Figure 1).	1
week 10	June 25th	July 1st	● Define the rest visConfig with: requiresUpdateStatus with all status params; default request/response handlers; one of categoryAxes on top (month) of the chart with labels truncating 3 characters (\`X-Axis\` can accept \`Date Range\` or \`Date Histogram\` within a year), the other categoryAxes (weekdays) on left of the chart; axisFormatter function in in labels field of categoryAxes to display the month text (Jan, Feb, etc) or weekday text (Sunday, Monday, etc); ● Call the calendar visualization and check the rendering chart and adjust any vis params.	1
week 11	July 2nd	July 8th	● Then, I can test the editor along with the calendar visualization. ● Run functional tests within the visualize app using Mocha, reuse some service like \`retry\`, \`log\`, \`pageObjects\` with changing editor buckets (\`X-Axis\`, \`Y-Axis\`, \`Split Chart\`) to see whether the months, weekdays and years rendered correctly.	1
Evaulations week 12	July 9th	July 15th	● Write or submit evaluations for mentors	2

week 13	July 16th	July 22nd	● With functional tests, test sample data with changing editor Metrics (aggFilters), Options to see the rendering charts.	2
week 14	July 23nd	July 29th	● Test the visualization on \`save\`, \`share\` and \`refresh\`, filtering conditions (in function tests, use Mocha to save and load calendar heatmap, examine whether it displays the correct chart by taking data screenshots). ● Test the calendar visualization interactions with \`Dashboard\`, (e.g. canvas chart resize rendering, data tooltip rendering, charts saving, editing, etc.)	1
week 15	July 30th	Aug 6th	● Test the calendar visualization both on \`Visualization\` and \`Dashboard\` with large amounts of data (functional tests within the visualize app using Mocha, reuse some service like \`retry\`, \`log\`, \`pageObjects\`). ● Clone the elastic/docs repo as a sibling of my kibana repo, document codes and evaluate git commits.	1
Code submission and Final Evaulations week 16	Aug 6th		● Submit final codes, push to remote master branch as well as create a pull request to the kibana master branch.	1
week 17		Aug 14th	● The end of the journey	

Why you?

For Kibana, I’ve submitted a pull request for a new feature of adding a show/hide button for filter bar component that can be used in various places (Discover, Visualize, Dashboard):
https://github.com/elastic/kibana/pull/17161#event-1523858619
Technologies: AngularJS, Mocha, Javascript, Less, HTML

Appendix

My thoughts on the calendar heatmap chart design are inspired by Google Charts (Figure 1) that render the year upfront along with some delimiting lines between months to give a better view of data

(But the Y categoryAxes need to have full text, e.g. Sunday, Monday to give a better view). When the Date Range or Date Histogram spans across multiple years, the visualization will render multiple charts with corresponding results (Figure 2).

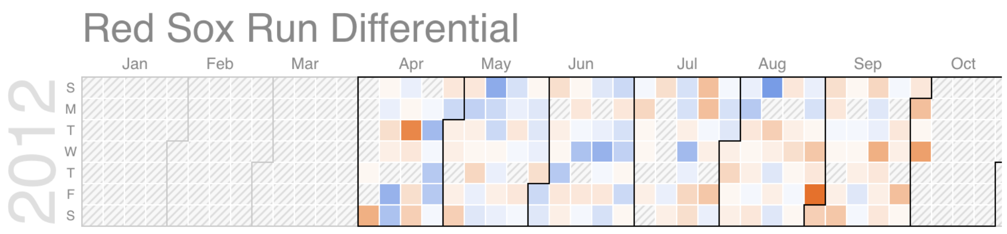


Figure 1. Google Charts - Calendar Visualization

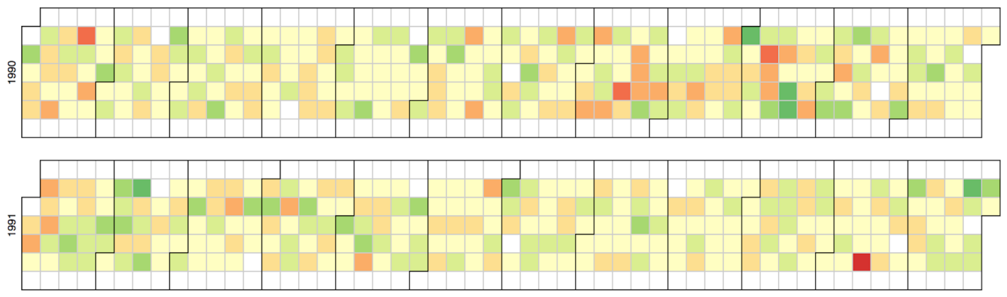
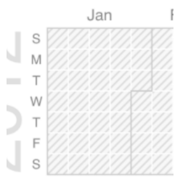


Figure 2. D3 Calendar Visualization

However, the canvas has defined clipPath that might prevents visualizations from rendering in whole (if the year span is too long). In addition, if there are too many calendar visualizations in a limited window size display, the size of each cell would be too small to observe. Thus, the limit for the number of charts in a single plot can be at maximum 2 (in case you need to compare the data in the same day from two specific years, like what is shown in Figure 2)

In the meantime, keep the X-Axis full with all days spans from the first day of a month to an end of the ending month even if the starting dates in index-pattern are in the middle of one of the month (**to keep the tick interval stable to make it easy in rendering**). For example, the data might span from 15th May to December 12th, the X-Axis must render from the first day of May to the end of December in that year.

Same for the Y-Axis to render the full weekdays from Sunday to Saturday. And keep the ignored portion of dates rendered in shady grey style like below: (to distinguish the regular cells (including white cells that are tracked) and ignored cells and no tooltip for these cells).



The main strategy is to fix X and Y scale to weekdays and month, then override the addSquares() method in heatmap along with some data preprocessing (padding some ignored data, the reason to process the data within chart drawing class is the **default behavior of aggregation filters, lucene filters should not change**) and specify a function to draw the month boundary strokes in dark black to achieve that goal (using `path` element of SVG with `d` variable to paint the stroke along the data cells edge (with `W`, `V`, ‘H’ and `Z` line commands to achieve that)).

References:

1. Google Charts, Calendar Chart:
   https://developers.google.com/chart/interactive/docs/gallery/calendar
2. D3 Calendar Visualization:
   https://bl.ocks.org/mbostock/4063318