Improving Functionality of the R Package plotly

Baobao Zhang

Application Title

Project Title: Improving Functionality of the R Package plotly

Project Short Title: Improving `plotly`

URL of Project Idea Page: https://github.com/rstats-gsoc/gsoc2015/wiki/ggplotly

Contact Information

Name: Baobao Zhang

E-mail: baobao.zhang@yale.edu

Phone Number: +1 (813) 368-9992

Link ID: baobaofzhang

Permanent Address: 4406 Oak Harbor Drive, Tampa, FL 33613 USA

Website: http://pantheon.yale.edu/~bz44/

GitHub: https://github.com/13bzhang/

IRC Nick: pandabz

Student Affiliation

Institution: Yale University, New Haven, CT USA (home institution); Massachusetts Institute of Technology, Cambridge, MA USA (pre-doc institution)

Program: PhD in political science, MA in statistics

Stage of completion: 2nd year

Contact to verify:

Frances Rosenbluth

Yale Political Science Director of Graduate Studies

frances.rosenbluth@yale.edu

Jay Emerson

Yale Statistics Director of Graduate Studies

john.emerson@yale.edu

About Me

My name is Baobao Zhang and I am a second-year PhD student in political science and a master's student in statistics at Yale University. I will be starting a pre-doc at MIT in summer 2015.

I have been coding in R for three years. My research has focused on non-parametric causal inference and experimental methods in the social science context. Data visualization has also been a daily part of my work. In the past, I have helped in the development and improvement of `big.data.frame`, an addition to the Big Memory Project, that supports larger-than-RAM data frame-like objects in R. In addition, I have worked on various other R-related projects, including creating a Raspberry Pi-based server that collects streaming Tweets using R and building a web-based riddle game that teaches students how to use R (much like the Python Challenge).

In addition to R, I have experience coding in Python and JavaScript. As a FOSS enthusiast, I have taught R and Python to students through my data consultant job at Yale's StatLab. As a part of the Open Access movement, I am currently working on a project to increase data transparency in clinical trial research. In my free time, when not modding drivers to run Linux smoothly on my Macbook, I am coming up with new projects for my Raspberry Pi.

Mentors

Mentor Names/E-mails: Chris Parmer chris@plot.ly and Marianne Corvellac marianne@plot.ly

I have been in touch with the mentors through e-mail. They have reviewed my application and given me feedback.

Overview

`plotly` is a R package whose function `ggplotly` converts graph objects made using `ggplot2` into a JSON format for display by Plotly, an interactive graphics web app. For my proposed project, I hope to increase the functionality of `plotly` by fixing bugs to allow greater customization of graphs and adding features that support all commonly used plot types (i.e., geoms) of `ggplot2`.

Motivation

`ggplotly` aims to serialize `ggplot2` graph objects in a standard format as JSON. Similar efforts have been applied to serialize visualizations into a standard JSON format for MATLAB and Python's `matplotlib`. A standard JSON representation of `ggplot2` graphs can be used to generate Plotly graphs, as well as other views of `ggplot2` graphs (e.g., animated and interactive graphics through the `animint` package). Furthermore, Plotly-rendered ggplots can be used as the interactive visualization layer for Shiny apps.

While `ggplotly` supports a large number of `ggplot2` features, there remains many that are not currently supported. Furthermore, plot types currently supported by `ggplotly` could have buggy features that hinder users' ability to customize plots.

As an applied statistician, I find Plotly to be a useful tool for users to share data visualizations in a transparent manner. Unlike traditional ways to share data visualization (e.g., via static images), plots published on Plotly allows access to the underlying data that generated the graphics. `ggplot2` has become one of the most popular data visualization packages. Improving the functionality of `ggplotly` would enable more R users, including students, academics, journalists, and businesses, to share their works via Plotly.

Detailed Project Goals

For my proposed project, I have two goals. First, I plan to fix bugs in existing features to improve plot customization. Second, I plan to add support for plot types that are not currently available. Items denoted by * indicate they might require changes made to the Plotly server.

Potential bugs to fix or features to improve depend on what open issues are not fixed and what bugs are unidentified at my starting date. Based on the current code, I see these features as needing improvement:

- legends: allow horizontal legends*; allow customized annotations; allow borders around legend symbols; allow multiple legends*
- ticks: allow irregularly spaced axes ticks*
- markers: allow manual sizing of markers; automate marker sizing so markers are not too large or small

The other goal is to help build an extensive ggplot2 graphing library, including graphics in the R Cookbook library and beyond, for testing ggplot2 graphs. Furthermore, I plan to incorporate the library into the ggplot2 visual test suite.

As part of the second goal, I plan to add support for some of the following plot types, or geoms:

- geom_bin2d: heatmap of 2d bin counts
- geom_crossbar: hollow bar with middle indicated by horizontal line
- geom_dotplot: dot plot*
- geom_freqpoly: frequency polygon
- geom_linerange: an interval represented by a vertical line
- geom_map: polygons from a reference map
- geom_pointrange: an interval represented by a vertical line, with a point in the middle
- geom_quantile: quantile lines from a quantile regression
- geom_raster: high-performance rectangular tiling
- geom_rug: marginal rug plots*
- geom_tile: tile plane with rectangles

For each of the plot types, I plan to write the code to support them, construct example plots as shown on the `ggplot2` documentation website, and conduct tests using `testthat`. As someone who has worked in spatial statistics, I am particularly interested in making sure that `ggplotly` can handle a wide range of map customization. I will discuss with my mentors which of these plot types should receive primary priority.

Tentative Timeline for the Project

If selected for Google Summer of Code 2015, I plan to follow this tentative schedule:

- Announcement of selected students – May 10 (end of my Spring Semester): get familiar with existing codebase; discuss with mentors what are primary and secondary features to be debugged or added

- May 10 – June 26: debug/add features of primary priority; the last week will be spent on writing documentation and constructing example plots

- June 26 – July 3: mid-term evaluation; discuss progress with mentors

- July 3 – July 10: make fixes as suggested by mentors in the mid-term evaluations

- July 10 – August 17: debug/add features of secondary priority; the last week will be spent on writing documentation and constructing example plots

- August 17 – August 28: final evaluation; discuss final fixes with mentors
- August 28 – September 15: clean up code as recommended by mentors; submit code

If thing are not going according to schedule, my contingency plan is to complete the tasks of primary priority and abandon the tasks of secondary priority.

Time Commitment

In the upcoming summer, I will have completed my coursework for my PhD in political science and master's in statistics. My primary responsibility will be to conduct original research at MIT's political science department. I am available to contribute 30 hours per week to the project.

Management of Coding Project

I will use GitHub to manage and submit my code. I will use `testthat` to test that my code correctly converts features of ggplot graph objects.

I plan to commit weekly. If I do not commit every week, that would indicate a problem.

Test

For my qualification test, I had to complete three tasks:

1. Easy: Use the `ggplotly` function to upload a plot made in `ggplot2` to Plotly.
2. Medium: Manually construct a Plotly JSON list in R and upload it to Plotly.
3. Hard: Write a `testthat` unit test for the salient features of the plot.

My code and results can be found in this GitHub repo: https://github.com/13bzhang/gsoc2015_plotly