**RIGHT: R Interactive Graphics via HTml**

Chung Ha SUNG

[Edit proposal](http://www.google-melange.com/gsoc/proposal/update/google/gsoc2013/sch8906/1)

**Email:** sch8906@gmail.com

**Short description:** This project aims to create R Interactive Graphics via HTml (RIGHT) package that enables interactive data visualization and analysis on a variety of platforms. RIGHT will provide a seamless analysis-visualization workflow in R and allow users to easily explore data and gain valuable insight for the subsequent analysis. For example, the user can highlight the relation among visual elements in multiple plots in a figure, or selectively remove outlier points and re-analyze the updated data either locally or remotely. Since HTML5 canvas and JavaScript will be used to create the visualization, it can be delivered to virtually any device/platform with a modern web browser. RIGHT will become a valuable tool for understanding large amounts of data, hence turning them into knowledge.

**Project title: RIGHT: R interactive Graphics via HTml**

**Project short title (30 characters): RIGHT**

URL of project idea page: <http://rwiki.sciviews.org/doku.php?id=developers:projects:gsoc2013:right>

**Abstract**

There is more and more emphasis on understanding data and turning them into knowledge as a larger amount of data become more readily available, and data visualization plays a critical role. It is usually the first thing done before any analysis or modeling is applied to the data, and it is also an important tool to communicate the results. As a result, many static visualization schemes are developed, and excellent software packages, including lattice and ggplot2 in R, exist to create them.  
Another important trend, I believe, is the widespread adoption of interactive visualization, which overcomes the limitations of conventional, static visualization. This trend is driven by diverse needs from users for easily exploring and understanding large multi-faceted data. The use of [D3 package](http://d3js.org/)  in New York Times articles and software products like [Google Chart Tools](https://developers.google.com/chart/)  and [Spotfire](http://spotfire.tibco.com/) are well-known examples. This trend will only accelerate as cheap, high-performance, and high-resolution computing devices, such as tablets, become more widely available.  
I propose to develop RIGHT, an R package that can deliver interactive data visualization to a variety of platforms. Since HTML5 canvas and JavaScript will be used to create the visualization, it can be delivered to virtually any device/platform with a modern web browser. There will be three main components to realize this: (1) HTML5 and JavaScript libraries for interactive graphics (RIGHTJS); (2) R API to set up the visualization; (3) functions for offloading computationally intensive downstream analysis to the server and updating graphics interactively on the client. More emphasis will be given on supporting data exploration than presentation: the package will highlight the relation among visual elements in multiple plots and facilitate downstream statistical analysis, which differentiates RIGHT from other tools such as Google Chart Tools and [googleVis](http://cran.r-project.org/web/packages/googleVis/index.html). RIGHT can also complement existing Qt-based ([cranvas](https://github.com/ggobi/cranvas" \t "_blank)) or SVG-based ([SVGAnnotation](https://github.com/duncantl/SVGAnnotation" \t "_blank), [gridSVG](http://sjp.co.nz/projects/gridsvg/), [SVGMapping](http://svgmapping.r-forge.r-project.org/SVG_Mapping/Home.html)) graphics packages by expanding the supported devices/platforms and by providing richer interactivity.  
Here is a YouTube video demonstrating a subset of the proposed functionalities, which I implemented as part of my senior project:

|  |
| --- |
| [https://lh6.googleusercontent.com/eCHvFwhHcnnbHlyyan9FDQd25e-N5lCoF5tsNLehxvoi-zPwIsh7__YvNELFIY66qAfMPOD_lnEgUxt8AyT8OFGVQBdgEJ4IimePjtEvb_rb7MX2hDbVBAf8](http://youtube.googleapis.com/v/thlwjYFC_yY?vq=hd1080;hd=1;autoplay=1) |

**CONTACT INFORMATION**

**Student name:**  Chungha SUNG

**Link\_id:**  sch8906

**Student postal address:** #305, 204dong Jungang Heights Ville 2 Danji Apt, Toegye-dong, Chuncheon-si, Gangwon-do 200-170, Korea.

**Telephone(s):** +82-10-5609-9607

**Email(s):**[sch8906 (at) gmail (dot) com](mailto:sch8906@gmail.com)

**Other communications channels:**  sch8906 in Skype/Google+.

**Student affiliation:**

**Institution:** Sungkyunkwan University (SKKU), Korea

**Program:** Bachelor of Science in Semiconductor Systems Engineering

**Stage of completion:** Graduating in August 2013

**Contact to verify:**

Prof. Jae W. Lee ([jaewlee (at) skku (dot) edu](mailto:jaewlee@skku.edu" \t "_blank))

Ms. Mina Oh ([minaoh (at) skku (dot) edu](mailto:minaoh@skku.edu" \t "_blank))

**Bio of Student:**

My name is Chungha SUNG, and I am a senior student in the Department of Semiconductor Systems Engineering at Sungkyunkwan University (SKKU) in Korea. I will receive my bachelor’s degree in August 2013. Last academic year, I received recognition on Dean’s List (awarded only to <5% of the student body at SKKU) and successfully finished a software engineering internship at AhnLab Inc., the largest anti-virus solution provider in Korea. I am very excited about this application for Google Summer of Code (GSoC). I have strong software development skills with C/C++, Javascript, and R, and am confident that I will make this project a great success.  
I started working on this project in January 2013, initially as a senior project with the two proposed mentors. I have since implemented the first prototype for the interactive graphics library using HTML5 and Javascript and successfully demonstrated its functionalities. Even with only a subset of the functionalities included in this proposal, my work is currently nominated for a department award for the *Best Senior Project of the Year*. I love the project and am confident about its value so much that I have decided to apply for GSoC. After this summer, I really look forward to seeing R users benefit from the technology I am developing!

**Schedule Conflicts:**

There is no scheduling conflict. I can fully concentrate on GSoC throughout the entire summer.

**MENTORS**

(this information is available on the project ideas pages)

**Mentor names:**  Prof. Jae W. Lee, Dr. Jung Hoon Lee

**Mentor emails:** [jaewlee (at) gmail (dot) com](mailto:jaewlee@gmail.com" \t "_blank), [flammy (at) gmail (dot) com](mailto:flammy@gmail.com" \t "_blank)

**Mentor link\_ids:** jaewlee, junghoon

**Have you been in touch with the mentors? When and how?**

* I have been working on a pilot project for this proposal with the mentors as a senior project since January 2013. I have held regular meetings every two weeks and many ad-hoc meetings with the mentors both in person and via conference calls. I have already prototyped a subset of the proposed functionalites to demonstrate their usefulness. You can find a demo video at [http://youtube.googleapis.com/v/thlwjYFC\_yY?vq=hd1080;hd=1;autoplay=1](http://youtube.googleapis.com/v/thlwjYFC_yY?vq=hd1080;hd=1;autoplay=1;), and the code is hosted at <https://code.google.com/p/r-interactive-graphics-via-html>.

**CODING PLAN & METHODS**

**Main steps & Planning:**

**HTML5 and JavaScript were chosen to drive the interactive visualization due to the following reasons:**

* Browsers natively support event handling necessary for interactive visualization, and JavaScript engines are powerful enough to visualize reasonably large data.
* The performance of web browsers (and JavaScript engines) will get only better, which is demanded by complex online services. RIGHT can piggy-back on the performance enhancements of JavaScript engines.
* HTML5 canvas and JavaScript make it easy to disseminate analysis and visualization results to wider audience since the web browser is the most widely adopted software platform from mobile devices to high-performance servers.

**This project consists of three main components:**

1. **HTML5 and JavaScript libraries (RIGHTJS) for interactive visualization**
   * This component implements various interactive graphics using HTML5 and JavaScript, which will be viewed with web browsers.
   * The performance of the library will be improved to handle a larger data set (e.g., data with 50K rows).
   * The code will be refactored to reveal the essential structure, and unit testing routines will be added. This can increase open-source community support and facilitate code maintenance and customization in the future.
   * The RIGHT library will support the following features:
     + Supported plots : scatter plot, bar plot (including histogram), box-and-whisker plot, line chart, pie chart.
     + Dataset uploading: Users can use local data set or upload their data set to the server.
     + Dataset processing: XML, instead of csv, will become the main format to exchange data, and csv (and JSON) will be supported.
     + Interactivity functions :
       - Tooltip box: Hovering cursor on a data point displays information about it.
       - Simultaneous selection: Corresponding data points across multiple charts are connected so that they will be selected at the same time. Multiple selection with ctrl/shift key and box selection is also supported.
       - Hiding: Selected data points can be hidden and not used for data processing.
       - Searching: Data points can be searched with a Boolean statement.
       - Table:  A table shows information about selected data points.
       - Right-click menu: The browser will show an context menu on right click.
2. **R API for convenient analysis-visualization workflow in R**
   * This component provides an intuitive R API to use RIGHTJS from R via code generation. As result, the users will have a leveraging R's strength in data manipulation and analysis. A Simple R API is prototyped and available in the code repository.
   * For GSoC, the R API will be refined for ease of use and expanded to accommodate new plots and support server-side offloading. Syntax of widely used graphics package, such as ggplot2, will be adopted.
3. **Server-side offloading of computationally intensive analysis and updating graphics interactively**
   * This component will be a valuable addition for weak client devices, such as phones and tablets, on which either R is not installed or running it may be too slow. For instance, if a user selects and hides (or removes) outliers in a plot and requests re-analysis, the analysis will be done on the server and the plots will be re-drawn on the client browser.
   * This feature will significantly improve interactivity on weak client devices, hence realizing my motto “Interactive visualization everywhere--from server clouds to mobile clients.”
   * A client-server communication protocol will be defined, and routines to support server-size computing will be implemented.

**Evaluation:**

**Three main aspects will be evaluated:**

1. **Functionality**
   * The RIGHT package will include, but not be limited to, the five types of interactive plots as detailed in the previous section.
   * Interactivity support functions will include, but not be limited to, multiple selection, interactive editing of data set (e.g., selecting, hiding, undoing, resetting), and tabular display of data. More consistent interactivity will be supported across all the five plot types.
   * Multiple popular browsers will be supported such as Chrome, FireFox, and IE.
2. **Performance**
   * The performance goal of RIGHT is to handle datasets with 50K+ entries while maintaining reasonable response time.
3. **Ease of Use**
   * To ensure ease of use and high quality of documentation, I plan to hire 5 “beta testers” who will test the functionality of RIGHT and provide feedback to improve its usability.

**Testing:**

**Three levels of testing will be performed to ensure** [**completeness of**](http://endic.naver.com/enkrIdiom.nhn?idiomId=9072873317614801b310e4415d57f2e9) **this project:**

1. **Unit test**
   * Unit test is very important for a successful open-source project, yet it is well known that it is tricky to apply the unit testing framework to GUI code. Therefore, I will try my best to clearly separate the “code that draws” from the rest, and sufficient unit testing code will be written for the non-drawing routines. I am planning to use [Qunit](http://qunitjs.com/) to setup the unit testing framework.
2. **Application level test(UI)**
   * Three problems will be used to test the overall functionality of the package.
   * Application test 1: 1000 randomly sampled rows from diamond dataset will be used to create 5 types of plots (scatter, histogram, box and whisker, line, and pie) to test layout and user interaction.
   * Application test 2: Outlier removal and interactive update will be tested using a linear regression problem.
   * Application test 3: Whether the RIGHT package can handle dataset with more than 50K rows will be tested on a web browser (Chrome or FireFox).
3. **User testing**
   * The R package will be tested by at least 5 testers in late August and early September to improve usability. Mobile platform, e.g. Android, will be tested to visualize the result.

**Documentation:**

The final R package will have the following documentations distributed through CRAN and YouTube.

* Standard R documentation for R API.
* Tutorials (vignettes) for R API using application testing problems 1 and 2.
* 5-minute tutorials on interactive graphics, server setup, interactive analysis, etc on YouTube.
* Demo video on YouTube.

**Perceived obstacles and critical objectives:**

* Functionality issues
  + R API to support server offloading for various types of analysis.
  + Browser compatibility.
* Performance issues
  + Handling 50K data sets for all supported plots without degrading interactivity (i.e., response time).
  + Server offloading without significantly increase the response time.

**TIMELINE:**

* **The main schedule will be divided by the following four parts.**
  + Prototype code cleanup and testing
  + Expand supported types of plots
  + Server offloading
  + R API improvement
* **Detailed schedule will be following.**

|  |  |  |
| --- | --- | --- |
| **Date** | **What to do** | **GSoC Timeline** |
| Week 1 (6/17~6/23) | - Refactor prototype code for plotting and R API.  - Begin server offloading part. | 6/17 : Students begin coding for their Google Summer of Code projects. |
| Week 2 (6/24~6/30) | - Implement unit testing for existing code and document the code. Unit testing and documentation for the rest of the code will be added as the code expands. |  |
| Week 3 (7/01~7/07) | Increase supported plot types. |  |
| Week 4 (7/08~7/14) | Expand and refine R API. |  |
| Week 5 (7/15~7/21) | Expand server offloading part. |  |
| Week 6 (7/22~7/28) | Testing and refinement for mid-term evaluation. |  |
| Week 7 (7/29~8/04) | - Testing and refinement for mid-term evaluation.   - Submit mid-term evaluations | 7/29 : Mentors and students can begin submitting mid-term evaluations.  8/02: Mid-term evaluations deadline. |
| Week 8 (8/05~8/11) | Improve server offloading performance. |  |
| Week 9 (8/12~8/18) | Refactor code to improve code quality. |  |
| Week 10 (8/19~8/25) | User testing and Improve R API. |  |
| Week 11 (8/26~9/01) | Refactor code to improve code quality. |  |
| Week 12 (9/02~9/08) | User testing and Improve R API |  |
| Week 13 (9/09~9/15) | Finalizing code and documentation. |  |
| Week 14 (9/16~9/22) | Finalizing code and documentation (pencils down). | 9/16: Suggested 'pencils down' `date. |

**- Contingency plan for things not going to schedule :**

  I reserved two weeks before the mid-term and final deadline for possible schedule slips. Please do not get alarmed that the plan for these weeks contains the words like “testings” and “documentation”; documentation and testing will be part of the on-going process throughout the duration of the project. I divided my plan into four parts, and I tried to lay out schedules for each part in detail as much as possible. I believe that the timelines are reasonable based on my experience with the pilot project, and delays in the schedule will be minimal.   
  Last but not least, as frequently communicating with my mentors, and I will maintain this policy during the summer. With such a regular mentoring and my previous experiences, I am confident that I will keep up with the schedule and successfully complete the project.

**MANAGEMENT OF CODING PROJECT**

**How do you propose to ensure code is submitted / tested?**

* The code will be regularly updated on Google Code repository (<https://code.google.com/p/r-interactive-graphics-via-html/>) into which I have already committed a prototype version. The code will be improved during the project and maintained afterwards. And, I will try to contact CRAN for getting regular feedback. Finally, the final package will be submitted to CRAN for the R community to use.

**What is the communication plan with mentors?**

* I can meet Prof. Lee any time in person. Weekly conference calls will be arranged with both of the mentors.

**Examples of similar coding problems that you have solved.**

* I worked on a pilot project of making R interactive graphs as part of my senior project.
* In addition, I took plenty of lectures in computer science, such as data structure, algorithm, computer network, operating systems, and computer architecture, etc. In these courses, I have done many software projects. For example, I made simple games using Matlab and C. I have good skills in C, Matlab, R, JavaScript, and HTML5.

**TEST**

**Problem 1:** Load Theoph dataset and plot concentration(conc) versus time (Time) by subject (Subject) with at least two graphics packages in R. Points for each subject should have different colors, and a legend should be added.

|  |
| --- |
| **(a) R code by using base graphics** |
| ## Script to create a sample R plot using the Theoph data set.  rm(list = ls())  cat("Clearing workspace...")    # Load data set:  data(Theoph)  set.seed(1)    ## Create plot:  subjectArray <- unique(Theoph$Subject)  numSubject <- length(subjectArray)    # Select color:  colorArray <- colors()[sample(length(colors()), numSubject, T)]    # Control the layout of the plots:  layout(matrix(c(1, 2), 1, 2))    plot(c(0, 1.1\*max(Theoph$Time)), c(0, 1.1\*max(Theoph$conc)), type = "n",      xlab = "Time", ylab = "conc")  for (iSubject in 1:numSubject) {     tempArray <- Theoph[Theoph$Subject == subjectArray[iSubject], ]   points(tempArray$Time, tempArray$conc, pch = 19, col = colorArray[iSubject])    } # for  legend("topright", legend = subjectArray, pch = 19, col = colorArray, cex = 0.8)    hist(Theoph$Time,      xlab = "Time", main = "") |
| **Result** |
| https://lh5.googleusercontent.com/r6Jf5vOTPJOHUm2SqvTwdAcm-VZ4JiqinlQfIHBgeMvvvFWo3Y_nf4ZEOeDrjMvts3StJ8qvraFhwwHAV0DyOLP4FlLkFp7lo61Mz4rEyPhCg79UKiA6VesJ |

|  |
| --- |
| **(b) R code by using ggplot2** |
| ## Script to create second prototype:    rm(list = ls())  cat("Clearing workspace...")    # Load data set:  data(Theoph)    # Create a layout within a single page:  grid.newpage()  pushViewport(viewport(layout = grid.layout(1, 2)))    obj.scatter <- ggplot(Theoph, aes(x = Time, y = conc, color = Subject)) + geom\_point()  obj.bar <- ggplot(Theoph, aes(x = Time, fill = Subject)) + geom\_bar(binwidth = 2, position = "stack")    print(obj.scatter, vp = viewport(layout.pos.row = 1,                                  layout.pos.col = 1))  print(obj.bar, vp = viewport(layout.pos.row = 1,                              layout.pos.col = 2)) |
| **Result** |
| https://lh5.googleusercontent.com/-UQP_DVXITDVT75HIBV_K5J2WKLN3sruJOEk3JHOl9k2SzkbfWL5vGvdJzGm1-LvjJQOcqrFdhpBD3nyWF3J81oeRUFuopF74xEkFXpkux6l8eOCvL2zTsGq |

**Problem 2:** Re-draw the plot created in problem 1 with JavaScript using the kineticJS library  (<http://kineticjs.com/>).

* I made scatter and histogram graphs, and more than two graphs are interactively visualized. They provide not only hovering and selection functions, but also box and multiple selection, searching box, and table functions.
* Code is here :  <https://code.google.com/p/r-interactive-graphics-via-html/>

|  |
| --- |
| **Result** |
| https://lh5.googleusercontent.com/-dqA3CrxSvR9TSrEldjauiMk_OFJOmUoTt9V4wpUpt_HQ5f0u7Y9Zj7u9fPtjdBNsvaXdiceI75DRh3XdEvXSGXJ7J1E7b_9na0DPAVszUtcnGVinuxUIDQe |