**RIGHT:** an *HTML* canvas and *JavaScript*-based interactive data visualization package for linked graphics

ChungHa Sung¹, TaeJoon Song², Jae W. Lee³, and Junghoon Lee³*

1. Sungkyunkwan University (SKKU), Suwon, Gyeonggi-Do, 440-746, South Korea
2. Samsung Electronics, Hwasung-Si, Gyeonggi-Do, 143-130, South Korea
*Contact author: jung_hoon.lee@merck.com

**Keywords:** interactive data visualization, linked graphics, *HTML* canvas, *JavaScript*

Interactive data visualization has received broad interest in the *R* community due to its obvious benefits over static visualization: more information can be delivered concisely and intuitively by user engagement. As a result, various *R* packages supporting single-layer, multi-layer, and linked graphics have been developed, including *rCharts*, *iPlots*, *cranvas*, *ggvis*, *animint* and *googleVis* [1]. *R* Interactive Graphics via *HTml* (*RIGHT, https://code.google.com/p/r-interactive-graphics-via-html/) is an interactive data visualization package for linked graphics based on *HTML* canvas and *JavaScript*. It provides an *R* API similar to base graphics to easily construct various interactive plots, including scatter, line, bar, pie, and box-whisker plots.

This poster presents an overview of *RIGHT* and the *JavaScript* data structure that enables linked graphics. *RIGHT* is the first package that implements linked graphs using *HTML* canvas and *JavaScript*. Linked graphics help answer obvious questions a collection of plots tend to raise: how one point in the plot is related to another point in another plot. *HTML* canvas and *JavaScript* make it possible to deliver the visualization to various platforms, including mobile devices, since they are standard web technologies supported by most modern web browsers (albeit some remaining compatibility issues). This approach can also benefit from the improvement of *JavaScript* performance every generation, driven by various web applications with ever increasing complexity and sophistication.

**References**

Abstract

R Interactive Graphics via HTML (RIGHT) is the first package that implements linked graphs using HTML canvas and JavaScript and supports efficient linked graphics that can show obvious relationship between multiple plots using same data. Also, HTML canvas and JavaScript make it possible to deliver the visualization to various platforms, including mobile devices since they are standard web technologies.

Motivation

Why linked graphs?

Link between two graphs can provide users with deep insight when analyzing the data large set. Clicking or dragging nodes from one graph just shows the related nodes of another graph.

How does work?

**<R command>**

```r
g{{{<code>
Obj <- RIGHT({
  plot(conc ~ Time, Theoph, color = color)
  runServer.RIGHT(loessArray, {
    obj <- loess(conc ~ Time, data = Theoph)
    xRange <- range(Theoph$conc)
    simArray <- data.frame(conc = seq(xRange[1], xRange[2], length.out = 100))
    simArray$time <- predict(obj, newdata = simArray)
    return(simArray)
  })
lines(loessArray, Time, conc)
hist(Time, Theoph)
}, Theoph)

Dot update
```

**<JavaScript>**

```javascript
<script>
  var Theoph = createMainStructureE(rawArr1);
</script>

```javascript
Object var
```javascript
  var xaxis = new Axis(1, Theoph, 'Time', 'conc', {legend: 'Subject'});
  var pointt = new Dot(xaxis, Theoph, 'Time', 'conc', {});
  var histObj = new ddply(Theoph, 'Time', 'Frequency', {});
  var hist = new Bar(xaxis, histObj, 'Time', 'Frequency', {});
  Theoph.draw();
  var AllAxisObjArr = [xaxis, xaxis2];
  eventTrigger(AllAxisObjArr);
</script>

```javascript
  // offload part.
  var loessArray = createMainStructureE('Theoph');
  var lofObj = new MakingLofObj(loessArray, 'Time', 'conc', {});
  var sF01 = new Line(xaxis, lofObj1, x1', x2', y1', y2', {});

  function() {
    setTimeout(function() {
      window.Shiny.onInputChange('Theoph', Theoph.$isHidden);
    }, 1);
  };
</script>

Future Work

This project is still under active development:
- To provide server-offload using Shiny for users to help them draw graphs on low-computing devices, such as mobile.
- To develop an intuitive API to make it easy to draw graphs (e.g., ggplot2-based API)

This work was supported in part by the Korean Ministry of Science, ICT & Future Planning (KEIT-10047038) and by Google Summer of Code (GSoC) 2013 and 2014.