LivelyR: Making R charts livelier

Aran Lunzer¹, Amelia McNamara^{2*}, Robert Krahn³

Viewpoints Research Institute
 University of California-Los Angeles
 SAP

*Contact author: amelia.mcnamara@stat.ucla.edu

Keywords: Visualization, statistics education, subjunctive interfaces

Traditional R graphics are static; if you want to modify an aspect of a graphic or the data preparation leading to it, you must edit and re-run your code. Even working in an interactive \mathbb{R} console, this process is laborious, and the resulting sequence of static outcomes is unhelpful for understanding what has changed from one to the next. Thus users are discouraged from parameter tinkering. For example, when you draw a histogram in R but don't specify a bin width, a default will be chosen for you. However, for a given dataset the choice of the bin width and the origin of binning can lead to dramatically different-looking histograms. If you don't tinker, you may not realise that your histogram is failing to reveal an important story in your data, or that the story it appears to tell is no more than an accident of the chosen bin divisions.

LivelyR builds on and extends **shiny** [3] and **ggvis** [4], which work together to bridge from R to the interaction possibilities of *Javascript* code running in a web browser. In our browsers we run Lively (in full Lively Web [1]), a full-fledged *Javascript* programming environment with rich facilities for building interfaces and for hooking into external applications (such as R) and web-page display abstractions (such as SVG and d3). LivelyR is, at this point, a research exploration into how a powerful combination such as this can be used to further enliven R charts by (a) turning rendered chart elements into interactive controls over the chart; and (b) providing built-in support for side-by-side comparison of alternative chart settings, based on subjunctive interface [2] techniques and a manipulable history of the user's interactions.

Unlike **ggvis**, all LivelyR execution is initiated and controlled from the *Javascript* side. From Lively we start an R process to act as a server that accepts fragments of R code for processing, and returns to Lively any textual results. A LivelyR web page uses this channel to start a **shiny** session and specify the dataset and desired charts; this session thereafter communicates with Lively through a web socket, using an extended form of the **ggvis** protocols.

In this talk we will demonstrate the latest version of LivelyR, showing various lively chart examples—including our approach to helping a user choose the parameters for a histogram—and will talk about where we imagine this type of system being useful. For example, we envisage using it to assist students in introductory statistics classes to understand the purpose of *R* functions' parameters, by making it easy to manipulate parameter values and see how the results are affected. Similar features could also be integrated into the charts in a newspaper article or academic paper, allowing a reader to adjust parameter values in the code and see how the authors' chosen values affect the analysis outcome. It would allow for a sort of code audit.

Acknowledgements: We thank Bret Victor for ideas and inspiration, and Dan Ingalls and Alan Kay for the freedom to explore.

References

- [1] Ingalls, D. and R. Krahn, et al. (2013). The Lively Web. http://lively-web.org/.
- [2] Lunzer, A. and K. Hornbæk (2010). Subjunctive interfaces for the Web. In A. Cypher, M. Dontcheva, T. Lau, and J. Nichols (Eds.), *No Code Required: Giving Users Tools to Transform the Web*, pp. 267–285. Morgan Kaufmann Publishers.
- [3] RStudio, Inc. (2013). shiny: Web Application Framework for R. R package version 0.8.0.
- [4] RStudio, Inc. (2014). ggvis: Interactive grammar of graphics for R. R package version 0.1.0.99.