Yet another graphics system for \textit{R}

Deepayan Sarkar\textsuperscript{1}\

1. Indian Statistical Institute, Delhi
*Contact author: deepayan.sarkar@gmail.com

\textbf{Keywords:} Visualization, Dynamic manipulation, Interaction

The traditional graphics model in \textit{R}, using a minimal common API targeting multiple backend “devices”, is flexible in some ways (especially for static graphics) but limited in others. Several alternative approaches have been developed in recent years to integrate other powerful visualization systems with \textit{R} (e.g., \texttt{rgl}, \texttt{rggobi}, iPlots eXtreme, \texttt{cranvas}). We present yet another approach with a somewhat different focus: efficient high-quality static graphics without the need to move to a different system for simple interaction. Our long-term goal is to implement high-level graphics systems similar to \texttt{lattice} or \texttt{ggplot2}, with support for multiple “backends” including but not limited to standard \textit{R} graphics devices, along with support for dynamic manipulation and interaction to the extent feasible depending on the backend.

We first describe a low-level framework implemented in the \textit{R} package \texttt{tessella}, with two essential elements. One element is a low-level API similar to that of \texttt{grid}, but of much more limited scope, which provides basic tools to manipulate viewport and layout-like objects. The other element is a set of abstract graphics primitives that may be implemented by multiple backends. The user (or package developer) writes graphics code using this common API, and the graphics output is rendered by a suitable backend chosen at runtime. We then describe a high-level system similar to \texttt{lattice} that is implemented using this low-level framework. We illustrate the system with examples that use backends implemented using standard \textit{R} graphics, Qt-based packages \texttt{qtbase} and \texttt{qtpaint}, and the Javascript canvas API.