Chapter 16
Scalable Vector Graphics

Abstract This chapter explores a powerful two-dimensional graphics format named SVG (scalable vector graphics), which is becoming widely used to represent object-based, vector (nonraster) graphics both on the Web and elsewhere. In addition to displaying scalable plots, the XML-based SVG format provides functionality for interactivity, animation, and a number of filtering effects that allow us to create rich graphical displays, and also to integrate them with Web pages. R can create static SVG displays, and we can post-process them within R as XML documents to add interaction and animation. We show how to provide simple interaction and animation with high-level R functions, and also how to add more customized interaction using R commands that use JavaScript at viewing time. We also discuss how to use both SVG and HTML to create rich “applications” and mash-ups in the Web browser.

16.1 Introduction: What Is SVG?

Graphical representations of data have significantly changed in recent years, where viewers expect to interact with a plot on the Web by clicking on it to get more information, produce a different view, or control an animation. Scalable Vector Graphics (SVG) [9] offers these capabilities; it is an XML format for describing two-dimensional (2D) graphical displays that also supports interactivity, animation, and filters for special effects.

Given that SVG is an XML grammar, SVG images can be constructed “from scratch” using the XML writing facilities described in Chapter 6. However, it is also possible to use R’s plotting facilities to create high-level, high-quality SVG graphics through the SVG graphics device (svg()) and the cairo rendering engine [7, 38]. Once created, we can post-process these SVG plots to add interactivity, such as hyperlinks, tool tips, linked plots, and arbitrary JavaScript code [10, 35] for handling events on user controls, e.g., sliders, buttons, and animation. This approach takes advantage of the standard plotting functions in R (including lattice and the traditional grz-model plotting functions) and leaves us to augment the resulting SVG plots, which we can accomplish through the XML parsing and writing facilities in the XML package [32].

A powerful aspect of SVG is that we can combine it with JavaScript to provide interaction and animation programmatically during the rendering of the SVG rather than declaratively through SVG elements. While this means using two programming languages, we note that the SVG plots are being displayed in a very different medium and use a language (JavaScript) that is widely used for Web content. These JavaScript additions can make the plot interactive for the viewer to control in a variety
of ways, e.g., linking points across scatter plots, animating a time series, and adding/removing subsets of data from a plot.

Similar to JPEG and PNG files, SVG documents can be included in HTML documents; they can also be in-lined within HTML content. Many commonly used Web browsers directly support SVG (Chrome, Firefox, Opera, Safari), and there is a plug-in for Internet Explorer. For example, Firefox can act as a rendering engine that interprets the vector description and draws the objects in the display within the HTML page. Quite differently from other image formats, SVG graphics, and their subelements, remain interactive when displayed within an HTML document and can participate as components in applications that interact with other graphics and HTML components. There are also non-Web-browser viewers for SVG such as Inkscape (http://www.inkscape.org/) and Squiggle (based on Apache’s Batik [2] http://xmlgraphics.apache.org/batik/tools/browser.html). Furthermore, SVG can be included in PDF documents via an XML-based page description language named Formatting Objects (FO) [23], which is used for high-quality typesetting of XML documents. For these reasons, SVG is a rich and viable alternative to the ubiquitous PNG and JPEG formats. While SVG may not be the most ideal approach to interactive graphics, it has many advantages that come from its simplicity and increasing use on the Web and in publishing generally.

Another advantage of SVG is that it uses a vector-based system that describes an image as a series of geometric shapes, in contrast to a raster representation that uses a rectangular array of pixels (picture elements) that represents what appears at each location in the display. That is, an SVG document includes the commands (in an XML format) to draw shapes at specific sets of coordinates, and these shapes are infinitely scalable because they are created from vector descriptions. The viewer can adjust and change the display, e.g., to zoom in and refocus, and maintain a clear picture.

In this chapter, we demonstrate some of the high-level functions for working with SVG documents to create interactive, animated displays from within R via the SVGAnnotation package [21]. As in other chapters, we first describe high-level functionality developed in SVGAnnotation that handles many of the common cases. In particular, we provide functionality to add tool tips and hyperlinks on plot components for many of the plots produced by the graphics functions in R. Next we delve deeper into the SVG grammar and some of the facilities it provides, and then demonstrate how we can use the package’s intermediate-level functions, which offer access to the components of the plot—e.g., point, axis, label, and title—in order to create arbitrary customized interactive graphical displays. The examples include linking points across scatter plots, controlling a spline smoother with a slider, animating points in a scatter plot, highlighting subsets of points using a choice menu, and using hyperlinks on a map to select HTML tables for display. Before describing these high-level and intermediate-level functions, we discuss the computational model behind them.

### 16.1.1 A Model for Adding Interactivity to SVG Plots

Our model for creating interactive plots uses the built-in R plotting tools to create the initial graphical display. (We plot to the SVG device available in R). Given the SVG tree, we augment (or annotate) the plot to include additional SVG elements and attributes and, possibly, JavaScript code. Since SVG is a grammar of XML, the graphical display created by the SVG graphics device in R is highly structured. This means it is relatively easy to examine and modify the SVG document programmatically. We use knowledge of the document structure to identify the various elements of an SVG plot/document that correspond to the components of the plot, e.g., point, axis label, title. Once identified, we augment the SVG document with information that enables interactivity and/or animation. In other words, there