D3 for R Users

Joyce Robbins

2019-08-07

Contents

Chapter 1

Introduction

This guide adapts Scott Murray's *Interactive Data Visualization for the Web, 2nd edition*–a required text for GR5702–for the needs of this course. Be sure to get the second edition, which is a comprehensive update to D3 version 4. The first edition uses D3 version 3, which is not compatible. (To add to the complication, the current version of D3 is v5. However, since differences between v4 and v5 are minimal, unless otherwise indicated in this guide, the code in *IDVW2* will work with either.)

We rely on the text heavily but also deviate from it in several ways. *IDVW2* is written for graphics designers not data science students so the pain points are somewhat different. As the title states, my intended audience is R users, though you certainly don't need to know R to use this resource.

In terms of content, we will use certain ES6 conventions not covered in *IDVW2* that make coding easier (and more like R!). We use different examples, though you are strongly encouraged to study Murray's code examples in addition to reading the text. Particularly through the first half, we don't follow the text in order, so always refer to this guide first which will direct you to the pages of the text that you should read.

Without further ado, let's start coding.

Chapter 2

Jump in the deep end

Let's skip the explanations and start coding in D3 right now. Why? So you can see the benefits and know what you're working toward when you get stuck in the weeds. Then we'll go back and start learning step by step.

2.1 Setup

If you don't have it already, download the Chrome browser.

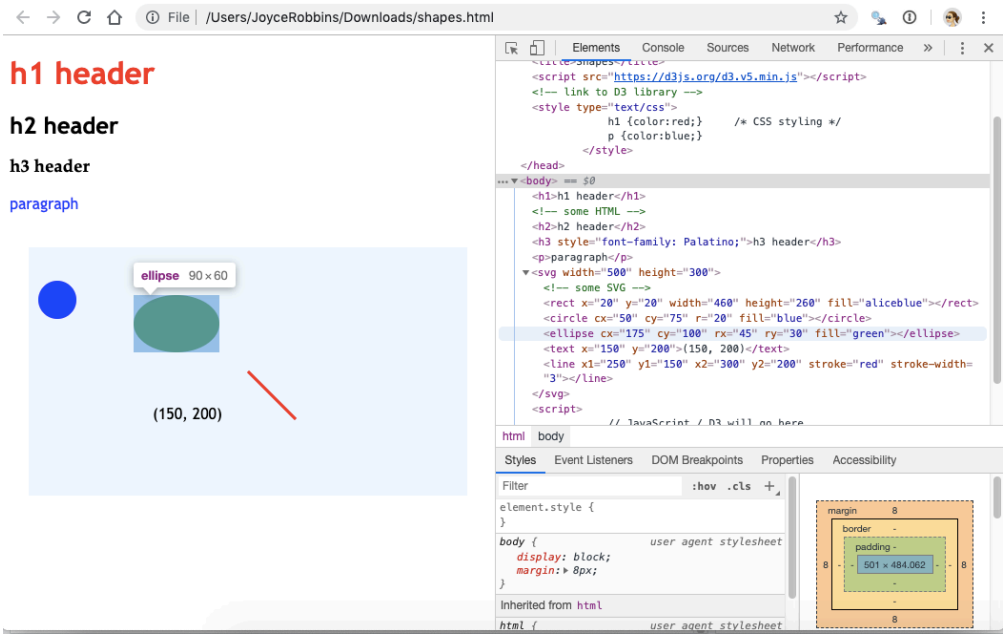
2.2 View the DOM in the Elements pane

1. First, you will need a *downloaded copy* of `shapes.html`. To download, open the following page and then click *File, Save Page As…*: `shapes.html`. (Or download a zip of the whole repo. Clicking here will start the download. Or fork and clone the repo).

2. If Chrome is your default browser, you can open `shapes.html` by double clicking it. Otherwise, open it with *File, Open File…* in Chrome.

3. Click *View, Developer, Developer Tools,* then the Elements tab.

4. Hover the mouse over various elements in the `<body> ... </body>` section. Observe the highlighted sections in the rendered web page on the left of the screen. Click on the mini black triangles to the left of the `<body>` and `<svg>` tags if needed to open these sections of the DOM tree. Your screen should look like this:



5. Now try the reverse: right click on elements on the web page, choose "Inspect" and see what is highlighted in the Elements pane. Get comfortable with the connection between the code on the right and the rendered elements on the left.

2.3 The JavaScript console

1. Switch to the Console tab, next to the Elements tab. Let's practice running some code (think R console.) Note that the code is unrelated to the `shapes.html` web page that we have open.

2. Type the following lines of code at the prompt (`>`), press enter after each line–that is, after the semicolon (`;`)–and see what happens:

```
3 + 4;

"3" + "4";

x = [1, 2, 3];

x[1];

x + 1;

y = {a: 3, b: 4};

y["b"];
```

2.4 Use D3 to change elements on the page

1. Now we'll start using D3 to manipulate elements on the page. Try the following, by entering one line at a time in the Console as before:

```
d3.select("circle").attr("cx", "200");

d3.select("circle").attr("cx", "500");

d3.select("circle").attr("cx", "100");

d3.select("circle").attr("r", "30");

d3.select("circle").attr("r", "130");

d3.select("circle").attr("r", "3");

d3.select("circle").attr("fill", "red");

d3.select("circle").attr("fill", "aliceblue");

d3.select("circle").attr("fill", "lightseagreen");
```

2. Refresh the page. What happened?
3. Go to Elements. Look at the value of the `y1` attribute of the SVG `<line>` element. Go back to the Console and enter the following:
```
d3.select("line").attr("y1", "10");
```
4. Switch back to Elements and observe. What happened?
5. Stay in Elements and refresh the page. What happened to `y1`?
6. Return to the Console to make style changes to the HTML elements:
```
d3.select("h1").style("color", "purple");

d3.select("h2").style("font-size", "50px");

d3.select("h2").style("font-family", "Impact");
```

2.5 Transitions

1. Try these:
```
d3.select("circle").transition().duration(2000).attr("cx", "400");

d3.select("ellipse").transition().duration(2000).attr("transform", "translate (400, 400)");

d3.select("line").transition().duration(2000).attr("x1", "400");

d3.select("line").transition().duration(2000).attr("y1", "250");

d3.select("p").transition().duration(2000).style("font-size", "72px");
```
2. Experiment with more transitions.

2.6 Interactivity

1. Set up a function to turn the fill color to yellow:

```
function goyellow() {d3.select(this).attr("fill", "yellow")};
```

2. Add an event listener to the circle that will be trigger a call to `goyellow()` on a mouseover:

```
d3.select("circle").on("mouseover", goyellow);
```

3. Test it out.
4. Add the same event listener to the ellipse. Test it out.
5. Create a function `goblue()` that changes the fill color to blue.
6. Add event listeners to the circle and ellipse that will trigger a call to `goblue()` on a *mouseout*. Test out your code.
7. Try out a click event. (Note the use of an anonymous function.)

```
d3.select("line").on("click", function()
  {d3.select(this).attr("stroke-width", "10");});
```

8. Try another click event. What's happening?

```
d3.select("svg").on("click", function()
  {d3.select("text").text(`(${d3.mouse(this)})`)});
```

Ok, now that we have a taste for what D3 can do, let's break it down, which we'll do in the next chapter.

Chapter 3

Web tech

Read: Chapter 3 "Technology Fundamentals" (pp. 17-62)

There is a lot of material in this chapter. It is worth making the effort to learn it now and start D3 with a solid foundation of elementary HTML/CSS/SVG/JavaScript.

Here we examine `shapes.html` from Chapter 1 to see how the various technologies are combined into a single document.

3.1 HTML – HyperText Markup Language

Note that `shapes.html` has an HTML extension; HTML in fact provides the structure for the document. It has a `<head>` and `<body>` section.

In the `<head>` section we use `<script>` tags to link to the D3 library:

```
<script src="https://d3js.org/d3.v5.min.js"></script>`
```

3.2 CSS – Cascading Style Sheets

CSS is used for styling web pages, and more importantly for our purposes, selecting elements on a page or in a graphic. `shapes.html` has style information in the `<head>` section that uses CSS:

```
<style type="text/css">
    h1 {color:red;}     /* CSS styling */
    p {color:blue;}
</style>
```

Here we specify that all HTML `<h1>` headers should be red and all HTML paragraphs `<p>` should be blue. This is an example of an *internal style sheet*. Later we will consider alternatives: *external style sheets* and *inline styling*.

3.3 SVG – Scalable Vector Graphics

SVG is a human readable graphics format that facilitates manipulation of individual elements. You may be familiar with `.svg` files. Here we have SVG graphics within `<svg>` tags in the `<body>` section of the HTML document:

```
<svg width="500" height="300">  <!-- some SVG -->
    <rect x="20" y="20" width="460" height="260" fill="aliceblue"></rect>
    <circle cx="50" cy="75" r="20" fill="blue"></circle>
    <ellipse cx="175" cy="100" rx="45" ry="30" fill="green"></ellipse>
    <text x="150" y="200">(150, 200)</text>
    <line x1="250" y1="150" x2="300" y2="200" stroke="red" stroke-width="3"></line>
</svg>
```

Rendered:

<rect x="20" y="20" width="460" height="260" fill="aliceblue"></rect>
<circle cx="50" cy="75" r="20" fill="blue"></circle>
<ellipse cx="175" cy="100" rx="45" ry="30" fill="green"></ellipse>
<text x="150" y="200">(150, 200)</text>
<line x1="250" y1="150" x2="300" y2="200" stroke="red" stroke-width="3"></line>

There are very few SVG tags that you'll need to know, and once we get going with D3, you will not have to code any SVG manually. It is worth doing a little to become familiar with the format and in particular to get used to the new location of the origin.

3.4 JavaScript

JavaScript is the most common language for making web pages interactive. Code is executed when page is opened or refreshed. So far we have run JavaScript in the Console, but have not included it in the web page itself. When we do so, it will be between `<script>` tags in the `<body>` section of the HTML document.

3.5 D3 – Data Driven Documents

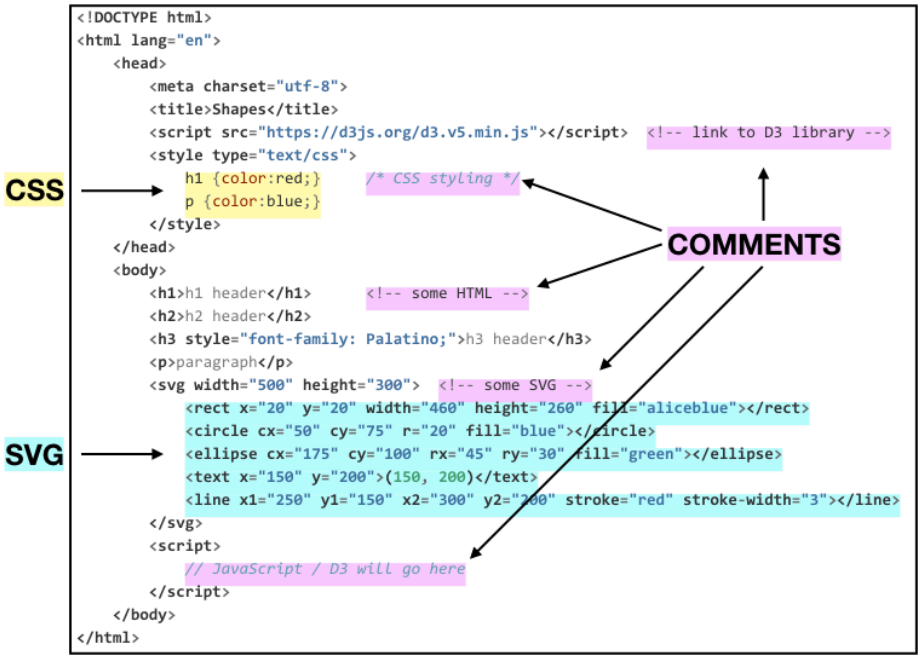
D3 is a JavaScript library well suited to interactive graphics. As such, it is also included between `<script>` tags in the `<body>` section. For D3 to work, you must link to the D3 library in the `<head>` section of the document.

There seems to be a misconception that D3 is a high level language. It is not. You will be working on the pixel level to create graphics, including drawing your own axes and doing other things that you're not used to doing if you've been working in R or Python. (On the bright side, after D3, R base graphics will seem very high level.)

It is legitimate to ask why you need to know D3 as a data scientist. Many if not most of you will not be coding in JavaScript from the ground up in your future careers. However, it's a great way to learn how interactive graphics work under the hood, and will give you a solid foundation which you can draw on to tweak visualizations that you build with high level tools such as Plotly.

3.6 Putting it all together

While `shapes.html` appears as a single consistent document, it is actually comprised of multiple languages. HTML, CSS, and SVG are already there, and we will be adding JavaScript / D3 soon.



Of note:

- An HTML is composed of lines or sections set off with tags. In particular `<style> ... </style>`, `<svg> ... </svg>`, and `<script> ... </script>` indicate the inclusion of CSS, SVG, and JavaScript/D3 respectively.
- For D3 to work, you must link to a D3 library. Here we link to an online version, but you can also download a copy from https://d3js.org and reference the local copy with:

```
<script src="d3.js"></script>
```

- Comment syntax varies with language:
  - `<!-- single or multiline HTML or SVG comment -->`
  - `/* single or multiline CSS comment */`
  - `// single line JavaScript comment`
  - `/* JavaScript multiline comment */`

Chapter 4

Back to D3

Some *significant* applications are demonstrated in this chapter.

4.1 Example one

4.2 Example two

Chapter 5

Final Words

We have finished a nice book.

Chapter 6

Scales