Introduction to the D3 library

Presenter Barnabé Monnot

Contents

› The anatomy of a webpage
› First D3.js script for SVG manipulation
› Some cool layouts
D3 is awesome for...

➤ Creating interactive data visualization

➤ Tying the visualizations with maps, geographical data

➤ Manipulating webpages à la jQuery

➤ With D3 available on Node.js, access to cool data manipulation procedures
Some examples
Some examples
Some examples
Some examples
Typical workflow

R → Get the data → D3 → Get the SVG → PS, AI
Mike Bostock

- D3 was created by Mike Bostock
- Evolved from the subject of his PhD thesis, working on a language, *Protovis*, to visualize data
- He is now in charge of making great interactive data visualizations for the New York Times (example, example)
First example: Bar chart

- Nothing fancy, but introduces key concepts such as:
  - Drawing in a webpage
  - Selections
  - Data-binding
  - Enter/Update/Exit pattern
  - Scales
First example: Bar chart

» Let’s create a JSFiddle, go to jsfiddle.net
First example: Bar chart

▷ What we want
First example: Bar chart

What we want

My eyes are burning!
First example: Bar chart

› What we want

Notice 10 px margin here

Top bar takes all the height
HTML first

- HTML is made up of tags, such as `<h1>` or `<div>`
- `<div>` is a particular one: defines a box in which to draw (text, image...)
- Tags can be given id’s (unique) and classes (reusable)
  
  ```html
  <h1 id='title'></h1>
  <p class='myp'></p>
  ```

- When we refer to them (in CSS or with D3), use `#` for titles and `.` for classes
  
  ```html
  #title // refers to the h1 title tagged #title
  .myp // refers to all elements classed .myp
  ```
D3 acts on selections of objects

d3.select("#title")
selects the DOM element named #title

Once we have selected an object, we can change its attributes

// Set content of h1 tag
d3.select("#title").text(“Introduction”)

We can also select all objects that have a certain property

// Select all elements classed bar
d3.selectAll(“.bar”)
var svg = d3.append("svg")
  .attr("width",200)
  .attr("height",100)

// creates <svg width=200 height=100>
SVG

```
// creates
<svg width=200 height=100>
  <rect x=50 y=40 height=20 width=40>
</svg>
```
We want to draw the data

```javascript
var data = [1, 3, 2, 4];
```

Each piece of data will be represented by a rectangle classed `.bar`

We select all `.bar` elements and attach the data to it

```javascript
var barsvg = d3.selectAll(".bar").data(data);
```
Enter selection

- We can get the `enter` selection via the `enter()` function applied to our selection `barsvg`.
  ```javascript
  barsvg.enter()
  ```

- The enter selection corresponds to the dashed boxes below: data that is *yet to be drawn*.
  ```javascript
  barsvg.enter()
  ```
  ![Diagram showing the enter selection with data 1, 3, 2, 4]
Scales

▷ Scales are an easy way to map inputs from a domain to outputs in a range

▷ There are a couple of cool options to work with: linear scales, logarithmic, qualitative...

```javascript
var scale = d3.scale.linear().domain([0,3]).range([10,100])
scale(1) = 40
scale(2) = 70
```
Exit selection

We can get the *exit* selection via the `exit()` function applied to our selection `barsvg`

`barsvg.exit()`

The exit selection corresponds to the dashed boxes below: data that is _yet to be removed from the drawing_

```javascript
barsvg.exit()
```

![Diagram showing exit selection with data points 3, 2, and 4]
Example: drawing a treemap

» D3 comes packaged with lots of nice layouts: force (network), dendogram (shown in the first slides) ...

» What this means is it computes boundaries for you, and your job is simply to draw them! => can then animate, make interactive ...

» Here we learn:

  » Using a layout

  » Color scales

  » Not drawing SVG (using standard HTML elements)
No more Fiddle

› We can’t use JSFiddle anymore: can’t load external resources in there...

› Fire up your favorite code editor (Notepad++, Sublime, Atom, TextMate, ...)

› Create a d3 folder somewhere (e.g, on your Desktop)

› Create a new file treemap.html
The HTML boilerplate

- We can’t use JSFiddle anymore, so we write a bit more HTML!

```html
<!DOCTYPE html>
<html>
	<meta>
		<title>D3 is awesome!</title>
		<script src="https://d3js.org/d3.v3.min.js" charset="utf-8"></script>
	</meta>
<body>
	// Where JSFiddle HTML was
</body>
</html>
```
The HTML boilerplate

▷ We add a script tag under the body one

```html
<!DOCTYPE html>
<html>
  <meta>
    <title>Malta</title>
    <script src="https://d3js.org/d3.v3.min.js" charset="utf-8"></script>
  </meta>
  <body>
    // Where JSFiddle HTML was
  </body>
  <script>
    // Your code goes here
  </script>
</html>
```
Getting the data

› Go to barnabemonnot.com/d3/data

› Decompress the zip in your D3 folder

› Open RStudio (yes! yes!)

› Use `setwd()` to set the directory to your D3 folder

› Remember last week? Use `dplyr`

› If you don’t have R, or I don’t have enough time, skip this part 😊
install.packages("dplyr")
library("dplyr")

dat <- read.csv("imports_manufactures.csv")
names <- read.csv("importkey.csv")
total <- dat %>%
  group_by(from) %>%
  summarise(sum(size)) %>%
  left_join(names, by=c("from"="id")) %>%
  select(name,2)

write.table(total, file="imports.csv", row.names=FALSE, col.names=FALSE, sep="","")
Example: drawing a map

 › D3 is also a breeze to work with maps.

 › Here we learn:

   » Loading external resources

   » Create groups in our SVG

   » Work with map data

   » Where Malta is

   » Respond to events and transitions
Example: drawing a map

› Go to barnabemonnot.com/d3/malta

› Get the code and paste it in a new file, *malta.html*, inside your D3 folder

› We are going to complete parts of it
Groups are nice to create layers to your drawing, easier then to reorder them if necessary

```xml
<svg width=200 height=100>
  <g class="borders"></g> // Borders are drawn first
  <g class="pois"></g> // Then the points of interest
  <g class="names"></g> // Then the names
</svg>
```