Herramientas de Visualización

Tema 5. D3.js. Escalando y dibujando ejes de un gráfico

Índice

Esquema

Ideas clave

5.1. Introducción y objetivos

5.2. Escala no ordinal o logarítmica

5.3. Ejes

5.4. Escala ordinal

5.5. Referencias bibliográficas

A fondo

Interactive Data Visualization for the Web

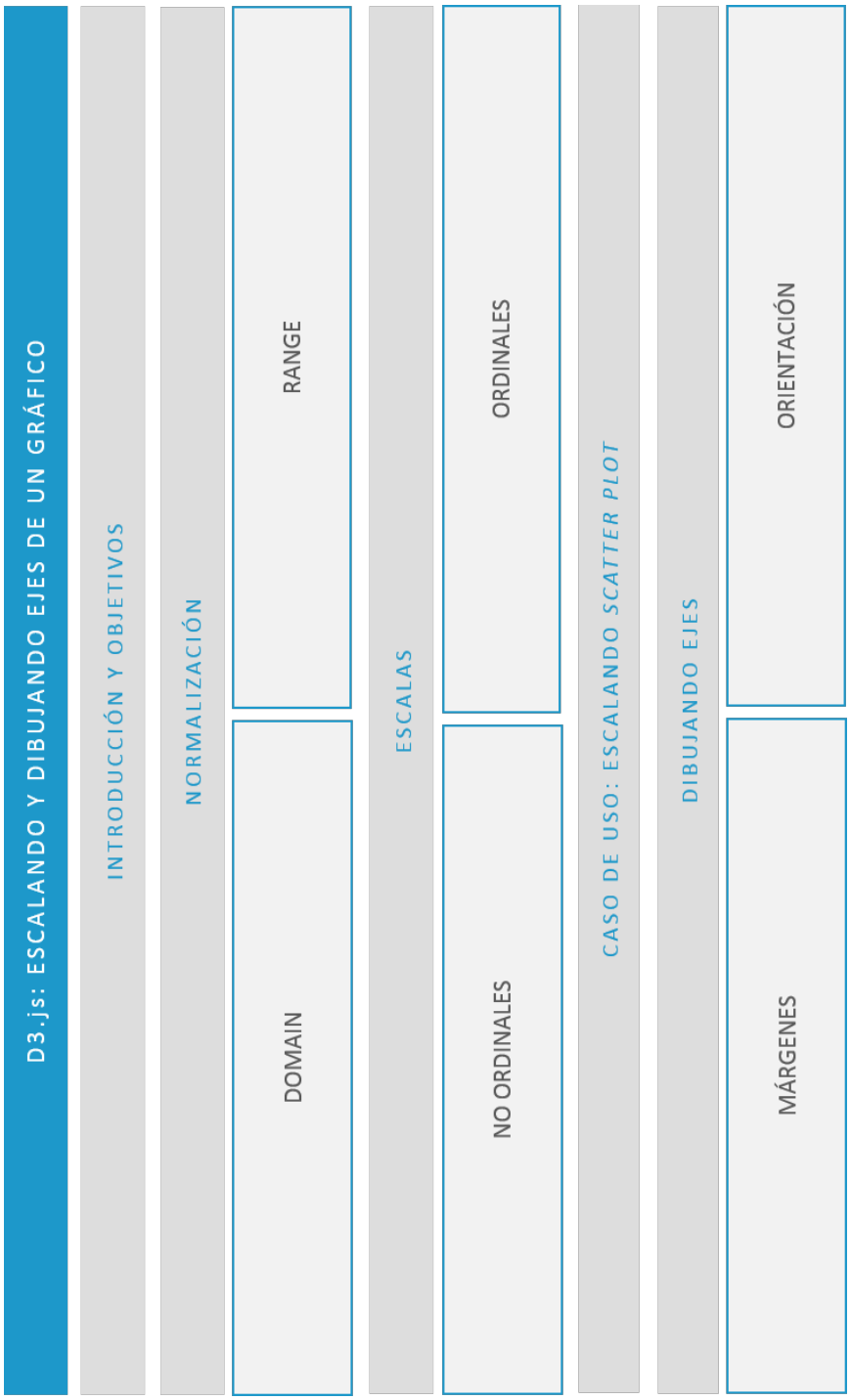
Setting up the Axes

Labelling multiple lines on a graph

How to rotate the text labels for the x Axis

Test

Esquema



5.1. Introducción y objetivos

Llegados a este punto, ya hemos podido ver que, en principio, no parece tan difícil dibujar en D3. Sabemos dibujar formas y jugar con los colores, que suele ser sinónimo de intensidad de los valores que estamos intentando visualizar o, simplemente, una forma para resaltar unos valores sobre otros. En este tema, intentaremos dar un paso más y trabajaremos con conceptos como las escalas.

Mike Bostock, uno de los principales desarrolladores de D3.js, define así las escalas:

> «Las escalas son funciones que relacionan un dominio de entrada con un rango de salida» (Coch, 2013).

Veremos qué significa esta definición. Este tema también se centrará en explicar cómo podemos crear esas funciones. Hemos de ser conscientes de la importancia de escalar una imagen, normalizar nuestros valores dará veracidad a nuestra visualización. Debemos conectar con nuestro usuario, convencerlo de lo que le queremos decir al escalar.

Finalmente, completaremos el gráfico, porque hasta ahora estábamos obviando una parte: los ejes.

«Los humanos ven lo que quieren ver» (Riordan, 2006)

Esta frase se ha escogido al considerar que la percepción humana es uno de los factores más importantes en las visualizaciones, las cuales las utilizaremos para validar nuestras hipótesis y permitir a otros validar las suyas. Pero como dice la frase, todos veremos lo que queremos ver. Definir una escala incorrecta dará una imagen incorrecta de nuestros datos y, por lo tanto, validará hipótesis incorrectas. No dejemos esa puerta abierta.

Ideas clave

Estudiaremos este tema al igual que los anteriores. Sigue las instrucciones como si fueran un tutorial, trata de entender todas las instrucciones del código y replica los pasos utilizando Brackets u otro editor de código. Es importante que practiques y pruebes pequeños cambios, lo que ayuda mucho a entender la lógica de D3.

5.2. Escala no ordinal o logarítmica

Primero, hemos de aclarar que trabajaremos con **escalas lineares no ordinales o logarítmicas**. Son las más típicas y las que utilizaremos por la facilidad de su comprensión.

En el libro que hemos visto en otros temas, *Interactive Data Visualization for the web*, de Scott Murray, se utiliza un muy buen ejemplo: manzanas y píxeles. Imaginémonos nuestro *dataset*, que es un *array* del valor de las manzanas que estamos vendiendo por mes.

Hasta ahora, si aplicábamos este concepto en las visualizaciones de tipo *bar charts*, el primer mes se visualizaba con la cantidad de 100px y el último con 500px. Sin embargo, si nuestro negocio de manzanas sigue creciendo a esta velocidad, en el mes doce ya tendremos que utilizar 1200 píxeles. Dependiendo del tamaño de nuestra visualización, seguramente estaremos fuera de rango. Por lo que introduciremos, a partir de ahora, dos nuevos conceptos: el ***input domain*** **y el** ***output range***:

- *Input range:* hace referencia a los valores que tenemos como **entrada**. Estos valores no podemos limitarlos, ya que es nuestra fuente de datos, que es quien nos da y escoge sus propios valores. En nuestro caso, son los valores de nuestro *array* .
- *Output range*: hace referencia a los **límites que ponemos nosotros** para mostrar los dos. Es decir, si decidimos que el valor mínimo son 10px y el mayor 350px nuestro *array* se mostrará en el gráfico siendo el valor 100 con 10px y el valor 500 con 100px. Aquí escogemos nosotros basándonos en el diseño de nuestra visualización.

Lo que estamos haciendo es similar al **proceso de normalización**: la normalización no deja de ser distribuir nuestros datos entre valores de 0 a 1 para poder trabajar con

todos nuestros datos en la misma escala.

Vamos a crear en D3 nuestra **escala**:

Ahora tenemos una función definida, pero no es muy útil porque le podemos pasar valores y siempre nos devolverá los mismos. No hemos definido ni el *input domain* ni el *output range*.

Si ejecutamos la sentencia anterior, obtenemos un 20 como resultado. Por lo tanto, para tener una función realmente útil, procederemos a **definir los dos valores** para nuestro *array* manzanas.

Recordad que también podemos **anidar las funciones**, por lo que todo lo podríamos hacer en una única sentencia:

Ahora tiene más sentido utilizar la función:

Si ejecutamos las anteriores sentencias, obtendremos 10, 180 y 350 como valores. Vamos a intentar aplicarlo a nuestro *Scatter plot* descrito en el tema anterior.

Recordemos nuestro *dataset*:

Como se ha mencionado antes, nosotros no debemos definir el *input domain*, ya que nos vendrá definido por los datos de entrada del *data source*. Antes ya habíamos definido el *input domain* manualmente, por lo que también introduciremos dos funciones:

¿Cómo las podríamos utilizar en nuestro *dataset*?

```
//Calcular el valor máximo de x y devuelve 480

d3.max(dataset, function(d) {

return d[0]; //Referencia al primer valor dentro de cada sub-array

});
```

Esto lo tenemos que hacer porque son *arrays* dentro de *arrays*. Miremos el siguiente ejemplo:

Si observamos la sintaxis de nuestra función, para devolver el máximo del primer valor de los *subarrays* en nuestro *dataset*, se parece mucho a cómo lo hacíamos para dibujar nuestro *scatter plot*:

```
.attr("cx", function(d) {

return d[0];

})

.attr("cy", function(d) {

return d[1];

})
```

Por lo tanto, es fácilmente deducible que si quisiéramos encontrar los máximos de nuestras "y", la **función de máximo** sería:

```
<span style="font-size: 12.8px;">d3.max(dataset, function(d) {

return d[1]; // Referencia al primer valor dentro de cada sub-
array

});</span>
```

Este patrón se repite en D3 habitualmente. Si ponemos todo junto, crearemos la siguiente función para el **eje X:**

```
<span>var xScale = d3.scale.linear()</span>
<span>.domain([0, d3.max(dataset, function(d) { return d[0]; })])
</span>
<span>.range([0, w]);</span>
```

En la función veremos dos aspectos: el mínimo lo hemos configurado a 0 directamente, pero también podríamos calcularlo con la función. Posiblemente lo que queramos es asegurarnos de que 0, en nuestro *input domain,* corresponde a 0 en nuestro *output range*, pero lo que es realmente relevante es el ***range***. Utilizamos w que era el width de nuestro SVG y definiremos algo muy similar para nuestro **eje de las Y**:

```
var yScale = d3.scale.linear()
.domain([0, d3.max(dataset, function(d) { return d[1]; })])
.range([0, h]);
```

Pero ahora, ¿qué código hemos de modificar en el *scatter plot*? Modificaremos el siguiente código:

```
.attr("cx", function(d) {
return d[0]; // Referencia al primer valor de cada sub-array
})
.attr("cy", function(d) {
return d[1];
})
```

Por este otro:

```
.attr("cx", function(d) {
return xScale(d[0]); // Referencia al primer valor de cada sub-
array
})
.attr("cy", function(d) {
return yScale(d[1]);
})    return xScale(d[0]); // Referencia al primer valor de cada
sub-array
```

Todavía tenemos un problema que arreglar. Seguramente habrás notado que los valores más altos de la y están en la parte baja del gráfico, mientras que los valores bajos de la y están en la parte más alta.

Ante esto, también modificaremos nuestra función yScale a:

```
var yScale = d3.scale.linear()
.domain([0, d3.max(dataset, function(d) { return d[1]; })])
.range([h, 0]);})
```

El código que resulta al final sería similar a este:

```
//Width and height
var w = 350;
var h = 200;
```

```
//Create SVG element
var svg = d3.select("body")
   .append("svg")
   .attr("width", w)
   .attr("height", h);
var dataset = [
[15, 80], [480, 90], [250, 50], [100, 73], [330, 95], [410, 120],
[475, 44], [25, 67], [85, 121], [220, 88]
];
var xScale = d3.scale.linear()
   .domain([0, d3.max(dataset, function(d) { return d[0]; })])
   .range([0, w]);
var yScale = d3.scale.linear()
   .domain([0, d3.max(dataset, function(d) { return d[1]; })])
   .range([h, 0]);
svg.selectAll("circle"
   .data(dataset)
   .enter()
   .append("circle")
   .attr("cx", function(d) {
return xScale (d[0]);
})
   .attr("cy", function(d) {
return yScale (d[1]);
})
   .attr("r", 5);})
```

Dando como resultado una gráfica perfectamente escalada al rango que hayamos definido, y de forma automática en función de los datos presentes en el *dataset*.

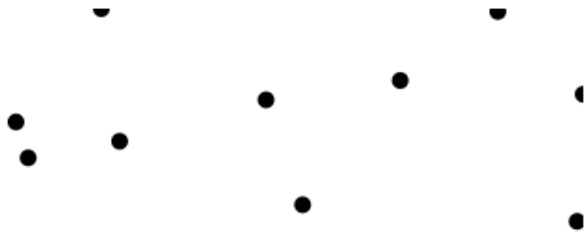


Figura 1. *Scatter plot* escalado a un rango definido.

Observarás que aquellos círculos que se sitúan en el límite del rango aparecen

cortados. Basta con **ajustar los márgenes** de nuestra gráfica.

Definimos el valor del margen y en este caso optamos por valores diferentes para los dos ejes:

Y lo aplicamos al rango:

```
var xScale = d3.scale.linear()
.domain([0, d3.max(dataset, function(d) { return d[0]; })])
.range([0+marginx, w-marginx]);
var yScale = d3.scale.linear()
.domain([0, d3.max(dataset, function(d) { return d[1]; })])
.range([h-marginy, 0+marginy]);
```

Figura 2. *Scatter Plot* escalado a un rango definido y con ajuste de márgenes.

5.3. Ejes

Es recomendable que resuelvas el ejercicio anterior antes de empezar este otro. Como siempre en D3, dibujar un elemento no es demasiado complicado:

Trabajaremos con una estructura simple de datos:

Tenemos que definir los valores del eje. Para ello le pasaremos la escala:

O todo junto:

```
var xAxis = d3.svg.axis()
   .scale(xScale)
   .orient("bottom");
```

Para dibujar el eje en sí necesitamos llamar a una función:

```
svg.append("g")
   .call(xAxis);
```

Veremos que el eje de la X nos aparece justo arriba, tal y como se muestra en la siguiente figura:

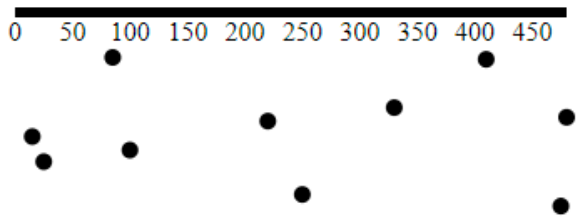


Figura 3. *Scatter Plot* con eje de la X en la parte superior.

Lo primero que haremos es intentar arreglar la **apariencia del eje** utilizando CSS. Le asignaremos el atributo class a nuestro elemento g y a nuestra clase CSS, la llamaremos axis.

```
svg.append("g")
   .attr("class", "axis") //Asignar la clase “axis"
   .call(xAxis);
```

Definiremos nuestro CSS:

```
.axis path,
.axis line {
fill: none;
stroke: black;
stroke-width:2;
shape-rendering: crispEdges;
}
.axis text {
font-family: sans-serif;
font-size: 11px;
}
```

Estéticamente queda mucho mejor:

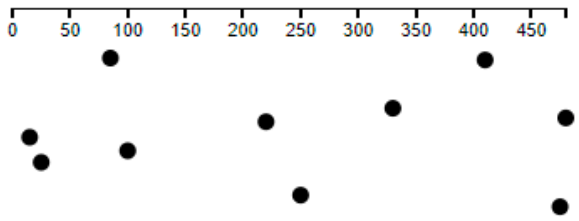


Figura 4. *Scatter Plot* y aplicación de estilo al eje de la X.

Presta atención a que no siempre los elementos CSS para los elementos SVG son los mismos que se pueden aplicar en elementos HTML normales. Vamos a resolver el problema que tenemos para que nuestro eje aparezca abajo.

Aplicaremos una transición al elemento:

```
var padding =30;
   svg.append("g")
   .attr("class", "axis")
```

```
    .attr("transform", "translate(0," + (h - padding) + ")")
    .call(xAxis);
```

La parte importante que resaltaremos es:

Aquí movemos nuestro elemento. Lo dejamos en x=0 y le movemos la «y» a la altura del svg, menos una cantidad que aquí hemos definido a 30. Y ahora el gráfico tiene el eje abajo.

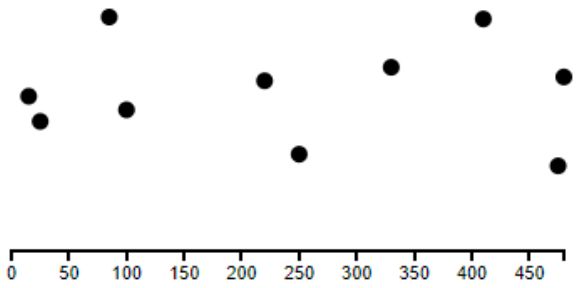


Figura 5. *Scatter Plot* con eje X situado en la parte inferior de la gráfica.

Podemos trabajar más con el eje X. Por ejemplo, si queremos limitar el número de divisiones en nuestro eje X, es decir, queremos tener solo cinco divisiones (0,100,200,300,400).

Lo definiremos al principio, al crear el eje de la siguiente manera:

```
var xAxis = d3.svg.axis()
    .scale(xScale)
    .orient("bottom")
    .ticks(5); // # de sticks
```

Nos da el siguiente resultado:

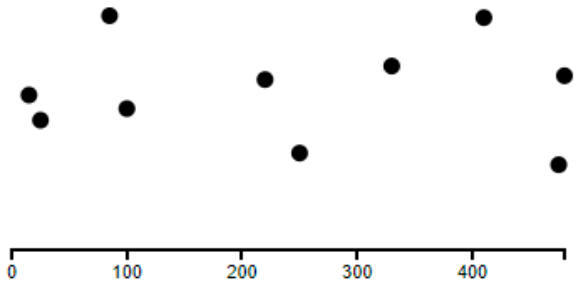


Figura 6. *Scatter Plot* con cinco divisiones.

Vamos a definir el eje Y:

```
var yAxis = d3.svg.axis()
   .scale(yScale)
   .orient("left")
   .ticks(5);
   svg.append("g")
   .attr("class", "axis")
   .attr("transform", "translate(" + padding + ",0)")
   .call(yAxis);
```

Nos daría el siguiente resultado:

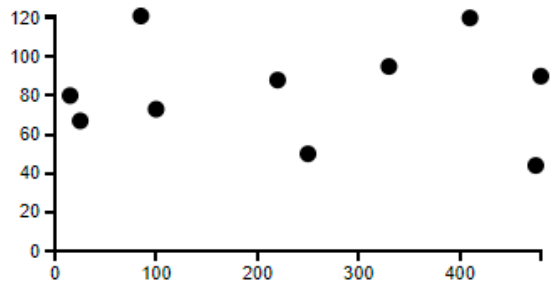


Figura 7. *Scatter Plot* con ejes X e Y.

5.4. Escala ordinal

Las escalas ordinales son diferentes a las lineares que se explicaron en el apartado anterior, donde los valores seguían una distribución continua.

¿Qué es una escala ordinal? Las escalas ordinales distribuyen los datos según **categorías** en lugar de una evolución lineal.

Por ejemplo:

- Notas al estilo americano: B, A, AA+.
- Categorías en los cuestionarios: totalmente en desacuerdo, desacuerdo, neutral, de acuerdo y totalmente de acuerdo.

Para ilustrarlo, empecemos con uno de nuestros gráficos de *bar charts*:

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="styles.css"/>
<script language="javascript" type="text/javascript"
   src="http://d3js.org/d3.v3.min.js"></script>
<script type="text/javascript">
function read(){
   //Width and height
   var w = 500;
   var h = 100;
   var barPadding = 1;
   var dataset = [ 5, 10, 13, 19, 21, 25, 22, 18, 15, 13, 11, 25];
   //Create SVG element
   var svg = d3.select("body")
      .append("svg")
      .attr("width", w)
      .attr("height", h);
   svg.selectAll("rect")
     .data(dataset)
     .enter()
```

```
      .append("rect")
      .attr("x", function(d, i) {
         return i * (w / dataset.length);
})
      .attr("y", function(d) {
         return h - (d * 4)
})
      .attr("width", w / dataset.length - barPadding)
      .attr("height", function(d) {
         return d * 4;
})
      .attr("fill", function(d) {
         return "rgb(0, 0, " + (d * 10) + ")";
});
svg.selectAll("text")
      .data(dataset)
      .enter()
      .append("text")
      .text(function(d) {
         return d;
})
      .attr("text-anchor", "middle")
      .attr("x", function(d, i) {
           return i * (w / dataset.length) + (w / dataset.length -
                  barPadding) / 2;
})
      .attr("y", function(d) {
          return h - (d * 4) + 14;
})
      .attr("font-family", "sans-serif")
      .attr("font-size", "11px")
      .attr("fill", "white");
}
</script>
</head>
<body onload="read()"></body>
</html>
```

Este código nos muestra el siguiente resultado:

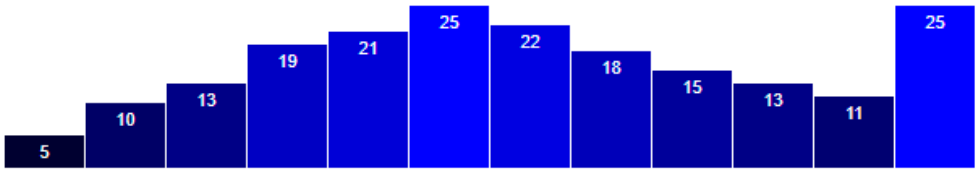


Figura 8. *Bar Chart*.

¿Qué podríamos hacer con las escalas ordinales? Introduzcamos en nuestro código las siguientes variables y definamos también las escalas e .

```
var w = 600;
var h = 250;
var xScale = d3.scale.ordinal()
.domain(d3.range(dataset.length))
.rangeRoundBands([0, w], 0.05);
var yScale = d3.scale.linear()
.domain([0, d3.max(dataset)])
.range([0, h]);;}
```

De forma similar a como lo hicimos en la gráfica de *scatter plot*, hemos de recordar **modificar nuestros X e Y**:

En las barras:

```
.attr("x", function(d, i) {
     return xScale(i);
     })
.attr("y", function(d) {
     return h - yScale(d);
     })
.attr("width", xScale.rangeBand())
.attr("height", function(d) {
     return yScale(d);
     })
```

Y en el texto:

```
.attr("x", function(d, i) {
     return xScale(i) + xScale.rangeBand() / 2;
     })
.attr("y", function(d) {
     return h - yScale(d) + 14;
     })
```

Lo importante de las últimas líneas de código es que hemos definido nuestro eje X con una escala ordinal en la variable .

```
var xScale = d3.scale.ordinal()
     .domain(d3.range(dataset.length))
     .rangeRoundBands([0, w], 0.05);  }
```

De esta forma nos aseguramos de que el orden en que se dibujarán las barras será el orden en el que está definido el *array*, fuente de los datos. ¿Cómo lo hacemos? Definiendo el dominio, pero utilizamos la función range. ¿Qué es lo que conseguimos con la función ?

En nuestro caso sería lo equivalente a aplicar esta sentencia:

Pero piensa que también podríamos generar un dominio tal que:

Una de las cosas positivas que tienen las escalas ordinales es que soportan las **funciones** . En lugar de utilizar valores continuos como en las escalas lineares, utilizamos valores discretos y los valores son determinados con anterioridad.

Para dividir nuestra escala en el dominio podemos utilizar la función , que divide nuestra escala en bandas basada en la longitud del dominio. Lo que hacemos es lo siguiente:

```
(w - 0) / xScale.domain().length
(600 - 0) / 12
600 / 12
```

```
50
```

Además, en el código de la visualización hemos incluido un pequeño espacio entre nuestras bandas para que estéticamente sean más claras con la función .

Como prueba, puedes comprobar en tu gráfico el comportamiento si le quitas el parámetro 0.05:

¿Cuál es la ventaja entre aplicar escalas lineares y las ordinales a efectos prácticos? Mucha mayor sencillez, lo que nos facilita un poco las matemáticas. Si antes teníamos esto:

Ahora tenemos:

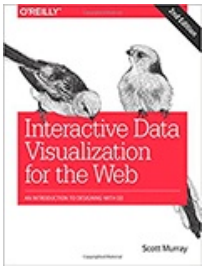
5.5. Referencias bibliográficas

Coch, G. (S. f.). *Escalas*. Tutorial de D3 en Español. http://gcoch.github.io/D3-tutorial/index.html

Riordan, R. (2006). *El ladrón del rayo*. Barcelona: Salamandra.

Interactive Data Visualization for the Web

Murray, S. (2017). *Interactive Data Visualization for the Web.* [S. l.]: O'Reilly.



Para este tema te recomiendo leer el capítulo 7 para tener más información sobre cómo escalar los dataset, y el capítulo 8 con más explicaciones sobre los ejes.

Setting up the Axes

Maclean, M. (S. f.). Setting up the Axes. En *Data Driven Documents D3.js: Tips and Tricks* (v.3.x). [S. l.]: Leanpub. Recuperado de https://leanpub.com/D3-Tips-and-Tricks/read#leanpub-auto-setting-up-the-axes

Para saber más sobre múltiples ejes, recomendamos la lectura de la sección del libro D3 Tips and Tricks dedicada a ello.

Labelling multiple lines on a graph

Maclean, M. (S. f.). Labelling multiple lines on a graph. En *Data Driven Documents D3.js: Tips and Tricks* (v.3.x). [S. l.]: Leanpub. Recuperado de https://leanpub.com/D3-Tips-and-Tricks/read#leanpub-auto-labelling-multiple-lines-on-a-graph

Para saber más sobre las gráficas como, por ejemplo, añadir una cuadrícula o un título, recomendamos la lectura de la sección del libro D3 Tips and Tricks dedicada a ello.

How to rotate the text labels for the x Axis

Maclean, M. (S. f.). How to rotate the text labels for the x Axis. En *Data Driven Documents D3.js: Tips and Tricks* (v.3.x). [S. l.]: Leanpub. Recuperado de https://leanpub.com/D3-Tips-and-Tricks/read#leanpub-auto-how-to-rotate-the-text-labels-for-the-x-axis

Para saber más sobre cómo rotar las etiquetas de los ejes, recomendamos la lectura de la sección del libro D3 Tips and Tricks dedicada a ello.

Test

1. ¿Qué es el *input domain*?

A. Es el rango de valores que nos da nuestra fuente de datos.

B. Es el dominio de datos que utilizamos para definir el título de una gráfica.

C. Es el rango de valores que definimos para nuestra escala.

D. Ninguna de las anteriores.

2. ¿Qué es el *output range*?

A. Es la leyenda de nuestra gráfica.

B. Es la escala que utilizaremos en nuestra gráfica.

C. Es el rango de valores que nos da nuestra fuente de datos.

D. Ninguna de las anteriores

3. ¿Cuál es el resultado al ejecutar estas sentencias?

```
var scale = d3.scale.linear();
scale(20)
```

A. 20.

B. 2.

C. 0,2.

D. 0,02

4. ¿Cuál es el resultado al ejecutar este código?

```
var scale = d3.scale.linear()
.domain([100, 500])
.range([10, 350]);
scale (100);
```

A. 100.

B. 10

C. 500

D. 350

5. ¿Cuál es el resultado al ejecutar este código?

```
var dataset = [
[5, 20], [480, 90], [250, 50], [100, 33], [330, 95],
[410, 12], [475, 44], [25, 67], [85, 21], [220, 88]
];
d3.max(dataset, function(d) {
return d[0]; //References first value in each sub-array
});
```

A. 480.

B. 495.

C. 95.

D. 12.

6. ¿Cuál es el resultado al ejecutar este código?

```
var dataset = [
[5, 20], [480, 90], [250, 50], [100, 33], [330, 95],
[410, 12], [475, 44], [25, 67], [85, 21], [220, 88]
];
d3.max(dataset, function(d) {
return d[1]; //References first value in each sub-array
});
```

A. 480.

B. 495.

C. 95.

D. 12.

7. ¿Qué es lo que hace "left" en el siguiente código?

```
var yAxis = d3.svg.axis()
.scale(yScale)
.orient("left")
.ticks(5);
```

A. Orienta el eje a la izquierda del eje Y.

B. Orienta los números del eje a la izquierda de la escala.

C. Orienta el eje a la izquierda del gráfico.

D. Orienta los números del eje a la izquierda.

8. ¿Qué es lo que hace este código?

```
var xAxis = d3.svg.axis()
.scale(xScale)
.orient("bottom")
.ticks(5); //Set rough # of ticks
```

A. Crea cinco divisiones en el eje X.

B. Crea seis divisiones en el eje X porque empiezan a contar desde 0.

C. Crea cinco divisiones aplicándolos a la escala.

D. Ninguna de las anteriores.

9. ¿Qué es lo que hace este código?

```
svg.append("g")
.attr("class", "axis")
.call(xAxis);
```

A. Asignar el atributo axis al elemento class.

B. Asignar el atributo axis al elemento xAxis.

C. Asignar el atributo class al elemento axis.

D. Asignar el atributo class al elemento g.

10. ¿Qué nos dibujará el siguiente código?

```
var yAxis = d3.svg.axis()
.scale(yScale)
.orient("left")
.ticks(5);
svg.append("g")
.attr("class", "axis")
.attr("transform", "translate(" + padding + ",0)")
.call(yAxis);
```

A. Un gráfica tipo *scatter plot*.

B. Los ejes X e Y de la gráfica.

C. Un gráfica con el eje Y.

D. El eje Y con las etiquetas orientadas a la izquierda.