

Maestría en Análisis y Visualización de Datos
Masivos

Herramientas de Visualización

Herramientas de Visualización

Tema 1. Introducción a las herramientas de visualización

Índice

[Esquema](#)

[Ideas clave](#)

[1.1. Introducción y objetivos](#)

[1.2. Librerías y herramientas de visualización](#)

[1.3. HTML, CSS y JavaScript](#)

[1.4. Editor código: Brackets](#)

[1.5. Referencias bibliográficas](#)

[Test](#)

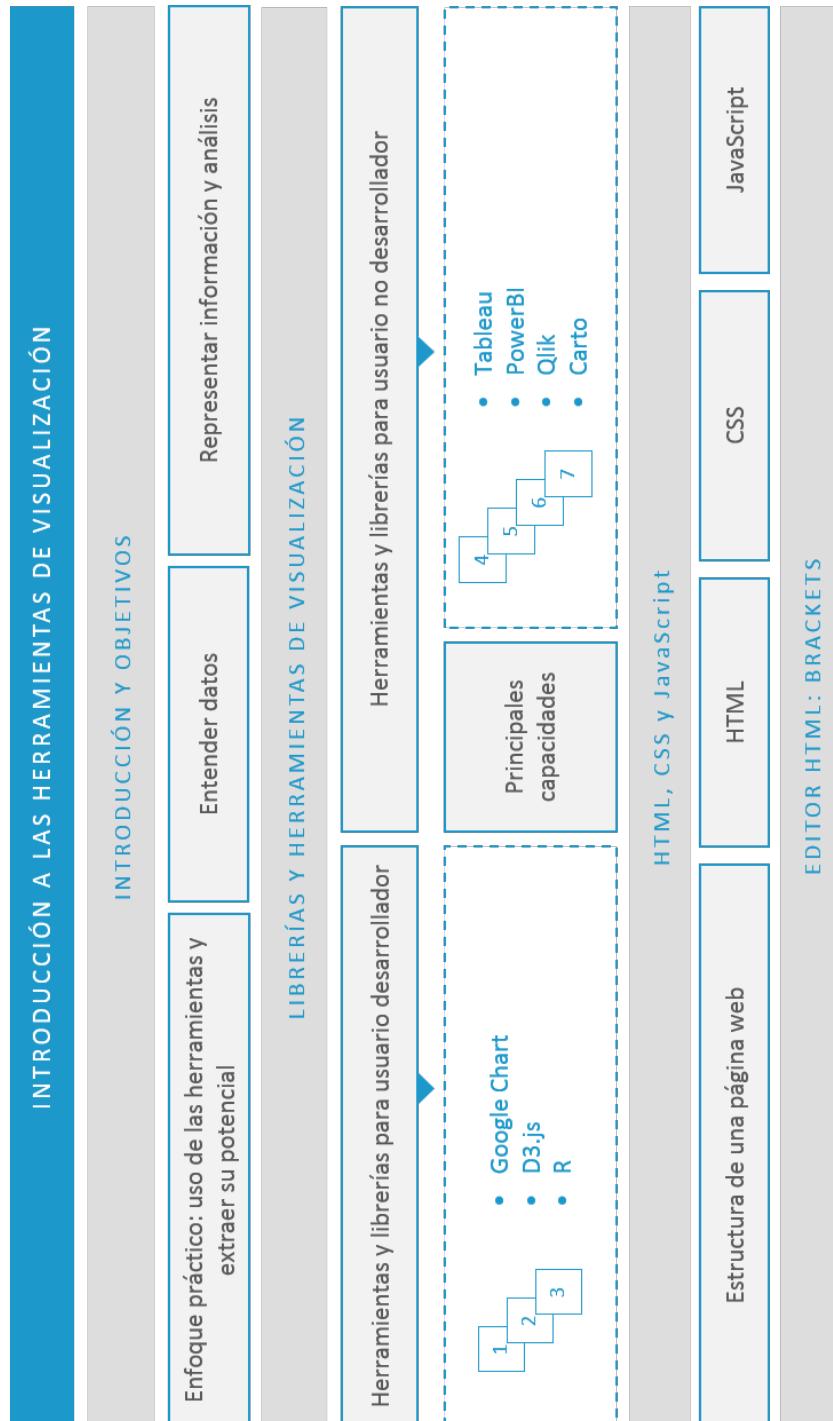
[A fondo](#)

[The 27 Worst Charts of All Time](#)

[Graphing guidelines](#)

[Pie charts: the bad, the worst and the ugly](#)

Esquema



1.1. Introducción y objetivos

El objetivo principal de este tema es que tengas claro cómo has de abordar esta asignatura de Herramientas de Visualización. Esta asignatura tiene un enfoque totalmente práctico en el que abordaremos diferentes librerías y herramientas de visualización y nos adentraremos en su uso y cómo extraer de ellas su mejor potencial.

Adicionalmente, para poder utilizar algunas de dichas herramientas es necesario conocer lenguajes de programación como HTML, CSS o JavaScript. Desarrollaremos los elementos necesarios para poder utilizarlos con solvencia.

En anteriores contenidos ya has tenido la oportunidad de conocer el lado teórico de las visualizaciones y, por lo tanto, aunque podemos discutirlo abiertamente, no se repetirá.

Queremos visualizar datos, explicar una historia con los datos, atraer la atención del lector con un mensaje, pero primero tienes que:

- ▶ Entender por ti mismo qué datos tienes.
- ▶ Ser capaz de representarlos gráficamente con las herramientas adecuadas.
- ▶ Comprobar que los destinatarios entiendan el mismo mensaje que pretendes explicar.

El principal consejo, respecto a la asignatura, es que tú mismo repases todo lo que se va explicando. Lleva su tiempo, pero si intentas de repente entenderlo todo, puedes tener problemas y tendrás que volver al principio. No por correr mucho llegarás antes al objetivo final.

Resumiendo, los **objetivos** de este tema son los siguientes:

- ▶ Familiarizarnos con la asignatura y su metodología.
- ▶ Identificar las herramientas que vamos a utilizar.
- ▶ Aproximarnos a HTML, CSS y JavaScript como lenguajes básicos que utilizaremos en algunas herramientas de visualización.

1.2. Librerías y herramientas de visualización

Los datos y sus relaciones no son siempre algo fácil de entender. Mientras que algunas personas pueden mirar una hoja de cálculo y encontrar intuitivamente la información que necesitan y sus relaciones dentro de una maraña de cifras, otras necesitamos un poco de ayuda, y ahí es donde la visualización de datos puede ser la solución.

La visualización puede ayudarnos a tratar con información más compleja.

No se trata de representar un conjunto de datos de forma amigable y con un diseño elaborado, sino también de hacerlo simplificando la complejidad y destacando aquella información que realmente es relevante, distinguiéndola del resto de los datos.

Una de las mejores formas de transmitir un mensaje es utilizar una visualización para llamar la atención rápidamente sobre los mensajes clave. Y al presentar los datos visualmente, también es posible descubrir patrones y elementos sorprendentes que no serían aparentes con la sola observación de los datos.

Hoy en día, existe una gran variedad de herramientas y recursos que nos permiten hacerlo. Así que para empezar a ayudarte hemos reunido algunas de las herramientas de visualización de datos más destacadas disponibles en la web.

Herramientas y librerías para usuario desarrollador

Para utilizar estas herramientas se requieren **conocimientos de programación**. Esta característica puede suponer una barrera de entrada, pero precisamente este inconveniente es también su mayor ventaja, ya que también les otorga enormes capacidades para personalizar gráficos y prácticamente infinitas posibilidades de

desarrollo.

Google Chart

Es la librería de visualizaciones utilizada por Google Spreadsheets, siendo la API de gráficos de Google. Se trata de una potente y sencilla librería, con una gran variedad de gráficos disponibles.

Accede a la página oficial de Google Chart:

<https://developers.google.com/chart>

D3.js

Es una librería de JavaScript que utiliza HTML, CSS y JavaScript para mostrar algunos diagramas y gráficos sorprendentes de una variedad de fuentes de datos. Esta librería, más que la mayoría, es capaz de algunas visualizaciones muy avanzadas con complejos conjuntos de datos. D3 tiene una poderosa capacidad para gestionar gráficos vectoriales en formato SVG (*Scalable Vector Graphics*).

Accede a la página oficial de D3.js:

<https://d3js.org/>



Figura 1. D3. Fuente: <https://d3js.org/>

R

¿Cuántas otras herramientas tienen un motor de búsqueda completo (<https://rseek.org/>) dedicado a ellas? R es un paquete estadístico enormemente potente, utilizado para analizar grandes conjuntos de datos, aunque con una curva de aprendizaje proporcional a sus elevadas capacidades. Esta librería requiere la utilización de Python como lenguaje de programación.

Accede a la página oficial de R:

<https://www.r-project.org/>

Herramientas y librerías para usuario no desarrollador

La **facilidad de uso** es la característica principal de este grupo de herramientas. Si algo las caracteriza es que es el propio usuario final quien tiene la capacidad para extraer y manejar los datos, crear informes o compartir los datos sin necesidad de pasar por un equipo de desarrollo, reduciendo de esta forma el tiempo que transcurre entre la necesidad y la respuesta. Evidentemente, la autonomía de los usuarios para la visualización de los datos con estas herramientas es máxima.

Según la consultora de mercado Gartner, en la actualidad, las principales plataformas modernas de analítica e inteligencia de negocio (*Analytics and business intelligence*, ABI) ya **no se diferencian por sus capacidades en la visualización de datos**. En su lugar, su diferenciación reside, por un lado, en cómo se integran con el resto de las aplicaciones corporativas de analítica e informes y, por otro, en sus capacidades aumentadas que les permiten acelerar las fases de preparación de los datos y generación de conocimiento para el negocio.

De forma resumida, Gartner evalúa todas las plataformas ABI con base en las siguientes 15 capacidades:

- ▶ **Seguridad:** capacidades que permiten la seguridad de la plataforma, la administración de los usuarios, la auditoría del acceso a la plataforma y la autenticación.
- ▶ **Capacidad de gestión:** capacidades para rastrear el uso, gestionar cómo se comparte la información y por quién, realizar análisis de impacto y trabajar con aplicaciones de terceros.
- ▶ **Cloud:** la capacidad de apoyar la construcción, despliegue y gestión de análisis y aplicaciones analíticas en la nube.
- ▶ **Conectividad de la fuente de datos:** capacidades que permiten a los usuarios conectarse e ingerir datos, estructurados y no estructurados, contenidos en varios tipos de plataformas de almacenamiento.
- ▶ **Preparación de datos:** soporte para la combinación de datos de diferentes fuentes, por medio de arrastrar y soltar, y realizable por los usuarios de negocio, así como la propia creación de modelos analíticos.
- ▶ **Complejidad del modelo:** soporte para modelos de datos complejos, incluyendo la capacidad de manejar múltiples tablas e interoperar con otras plataformas analíticas.
- ▶ **Catálogo:** la capacidad de generar y agregar automáticamente un catálogo que permita la búsqueda de los objetos creados y utilizados por la plataforma.
- ▶ **Insights automatizados:** un atributo fundamental de la analítica aumentada es la capacidad de aplicar técnicas de *machine learning* e inteligencia artificial para generar automáticamente conocimientos para los usuarios finales (por ejemplo, identificando los atributos más importantes de un conjunto de datos).

- ▶ **Análisis avanzado:** capacidades analíticas avanzadas a las que los usuarios pueden acceder fácilmente, ya sea porque estén contenidas en la propia plataforma ABI, o bien porque puedan utilizarse mediante la importación e integración de modelos desarrollados externamente.
- ▶ **Visualización de datos:** soporte para cuadros de mando interactivos y la exploración de datos mediante la manipulación de imágenes gráficas. Se incluye una serie de opciones de visualización que van más allá de las de los gráficos circulares, de barras y de líneas, como son los mapas de calor y de árboles, mapas geográficos, gráficos de dispersión y otros visuales de propósito especial.
- ▶ **Consulta de lenguaje natural:** permite a los usuarios consultar los datos utilizando búsquedas basadas en lenguaje natural.
- ▶ **Narración de datos:** capacidad de combinar la visualización interactiva de datos con técnicas narrativas a fin de empaquetar y ofrecer conocimientos de forma convincente y fácil de entender para su presentación a los responsables de negocio.
- ▶ **Análisis integrado:** las capacidades incluyen un SDK con API y soporte para estándares abiertos con el fin de integrar el contenido analítico en un proceso de negocio, una aplicación o un portal.
- ▶ **Generación de lenguaje natural (*Natural Language Generation*):** la creación automática de descripciones lingüísticamente ricas de los conocimientos encontrados en los datos.
- ▶ **Presentación de informes:** la capacidad de crear y distribuir a los usuarios informes de forma programada.

Tomando como referencia estos criterios de evaluación, Gartner elabora anualmente el informe denominado *Magic Quadrant for Analytics and Business Intelligence Platforms*, donde posiciona a las diferentes plataformas y herramientas del mercado.

Figure 1. Magic Quadrant for Analytics and Business Intelligence Platforms

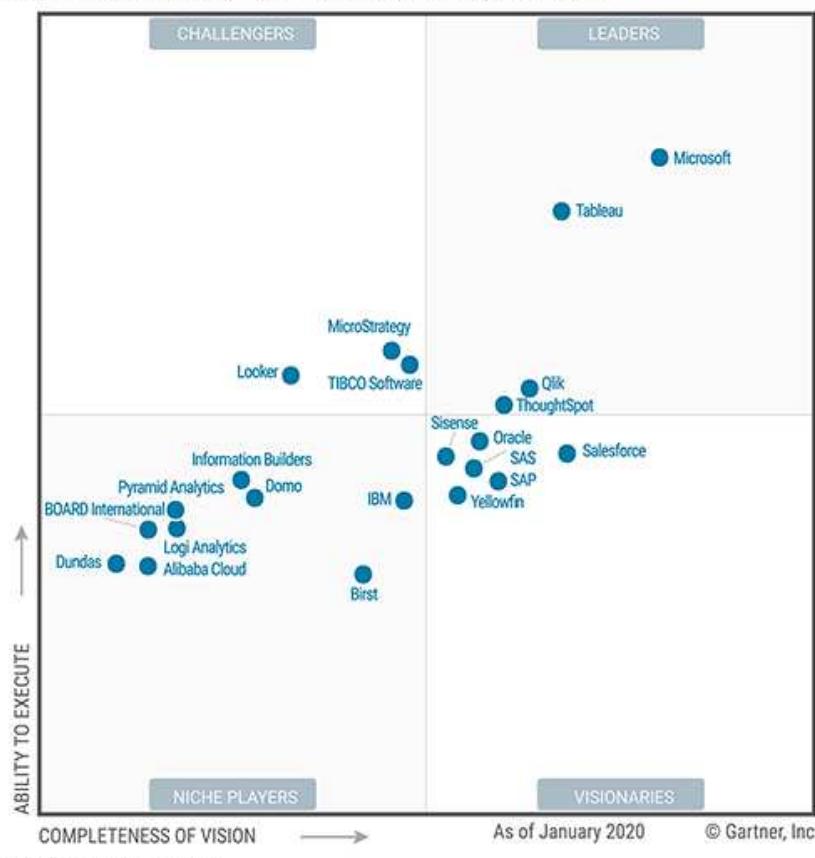


Figura 2. Cuadrante mágico de Gartner sobre plataformas de analítica e inteligencia de negocio. Fuente: <https://www.qlik.com/us/gartner-magic-quadrant-business-intelligence>

En la asignatura abordaremos aquellas herramientas consideradas líderes del mercado según Gartner: Microsoft (PowerBI), Tableau, Qlik... y adicionalmente Carto por su especialización en datos cartográficos.

Tableau

Repleto de gráficos, tablas y mapas, Tableau Public es una popular herramienta de visualización de datos. Los usuarios pueden arrastrar y soltar fácilmente los datos en el sistema y ver cómo se actualizan en tiempo real.

Accede a la página oficial de Tableau:

<https://www.tableau.com/>



Figura 3. Tableau. Fuente: <https://www.tableau.com/es-es>

PowerBI

Es la propuesta de Microsoft que nos permite conectar diferentes orígenes de datos, transformarlos, limpiarlos, relacionarlos y visualizarlos de forma eficiente y sencilla.

Accede a la página oficial de Power BI:

<https://powerbi.microsoft.com/>



Figura 4. Power BI. Fuente: <https://powerbi.microsoft.com/es-es/get-started/>

Qlik

Es una herramienta que se centra en ofrecer al usuario final la posibilidad de extraer y manipular directamente los datos, creando informes sencillos y todo esto sin la necesidad de pasar por un equipo de desarrollo. Es decir, maximiza la facilidad de uso para usuarios no desarrolladores.

Accede a la página oficial de Qlik:

<https://www.qlik.com/>

Carto

Es realmente una herramienta que hay que conocer. La facilidad con la que puedes combinar datos tabulares con mapas es insuperable. Por ejemplo, puedes alimentar un archivo CSV de cadenas de direcciones y lo convertirá en latitudes y longitudes, además, las trazará en un mapa.

Accede a la página oficial de Carto:

<https://carto.com/>

1.3. HTML, CSS y JavaScript

Esta es una asignatura completamente práctica, por lo que a la hora de utilizar las herramientas y librerías que requieren lenguajes de programación, asumiremos que sabes qué es HTML, CSS y JavaScript. En caso de que no tengas conocimiento previo, te recomiendo echar un vistazo a los siguientes tutoriales:

- ▶ HTML: <https://www.w3schools.com/html/>
- ▶ CSS: <https://www.w3schools.com/css/>
- ▶ JavaScript: <https://www.w3schools.com/js/>

No obstante, demos un repaso a la estructura de una página web y a los conceptos más habituales de estos tres lenguajes.

Estructura de una página web

Una página web se construye con base en los siguientes niveles:

- ▶ **Estructura y contenido:** donde se incluyen los elementos de la página que vemos en el navegador y que se componen de textos, imágenes, enlaces, párrafos, tablas, capas, etc.
 - El lenguaje a utilizar es HTML.
- ▶ **Apariencia:** donde se determinan los colores, tipografías, fondos, tamaños, alineación, etc. de todos los objetos presentes en la página web.
 - El lenguaje a utilizar es CSS.

- ▶ **Comportamiento:** aporta el grado de inteligencia necesaria a la página para poder mostrar efectos y animaciones, también realizar validaciones, automatizar tareas, conectar con bases de datos, llamar a APIs (interfaz de programación de aplicaciones), utilizar librerías externas, etc.
 - Existen multitud de lenguajes de programación que nos permiten trabajar tanto en el frontal como en el *backend*, pero a los efectos de su utilización con herramientas de visualización emplearemos JavaScript.

HTML

HTML es un lenguaje de marcas utilizado para el desarrollo de páginas web. Es una forma de codificar con base en **marcas o etiquetas** que contienen información adicional acerca de la estructura del texto o su presentación.

HTML define la estructura y contenido, pero no define su estilo visual.

Para eso se usará CSS.

Un **documento** HTML contiene dos partes principales: la cabecera y el cuerpo del documento.

- ▶ La cabecera (head) suele contener los metadatos del documento.
- ▶ El cuerpo (body) suele contener el contenido que se muestra en la página final. Si tenemos la necesidad de incluir documentos externos como archivos JavaScript y CSS, esto se hará en la cabecera.

Dispones de más información sobre la sección head en:

https://www.w3schools.com/html/html_head.asp

Si quieres profundizar en la sección body utiliza el siguiente enlace:

https://www.w3schools.com/tags/tag_body.asp

Si nunca has intentado trabajar con HTML, entra simplemente en esta dirección:

http://www.w3schools.com/html/tryit.asp?filename=tryhtml_intro

Cambia el contenido de los códigos HTML que aparecen y aprieta el botón de Run.

CSS

CSS nos permite generalizar y definir el estilo de la página HTML que se visualizará en el navegador y de cada elemento HTML de manera exacta. De esta forma, CSS aísla el contenido de la presentación.

De forma similar a JavaScript, el código se ha de introducir entre las *tags*: `<style>` y `</style>`. Seguiremos el ejemplo anterior, ¿qué pasa si añadimos al anterior ejemplo el siguiente CSS?

```
<style>
body{
background-color:#d0e4fe;
}
h2{color:orange;
text-align:center;
}
p{
font-family:"Times New Roman";
font-size:20px;
}
</style>
```

sobrescribirá el estilo del cuerpo de nuestro HTML. En este caso, estamos definiendo un nuevo color para el fondo de la página. Todos los colores tienen una representación hexadecimal. Si quieres saber a qué color corresponde ese código, edita tu HTML o introduce el código en la siguiente web: <http://html-color-codes.info/>

definirá el nuevo estilo de nuestra cabecera con un nuevo color y alineando el texto en el centro. Podemos observar que el color se define con el nombre del color en lugar del valor hexadecimal. Es una posibilidad para ciertos colores.

definirá el estilo de nuestra fecha. Aquí cambiamos el tipo y tamaño de fuente. Hay muchas más posibilidades que puedes explorar tú mismo. Este es un ejemplo muy básico.

Tanto JavaScript como CSS se pueden definir en otros archivos. Si los definimos en archivos externos, los tendremos que conectar con etiquetas dentro de la cabecera en nuestro HTML (lo que hemos marcado en negrita es lo deberíamos modificar):

- ▶ JavaScript:

- ▶ CSS:

En CSS existen 3 tipos de selectores o identificadores para saber sobre qué elemento HTML debemos aplicar el estilo:

- ▶ De **elemento** HTML: ...
- ▶ De **identificador**: todos los elementos HTML cuya propiedad **id** tenga un determinado valor, tendrán ese estilo.
- ▶ De **clase**: todos los elementos HTML cuya propiedad **class** tenga un determinado valor, tendrán ese estilo.

Dispones de multitud de ejemplos que te ayudarán en el uso de CSS en el siguiente enlace:

https://www.w3schools.com/css/css_examples.asp

JavaScript

JavaScript nos permite añadir un poco de interacción más avanzada. Se suele definir el contenido de las funciones dentro de dos etiquetas: `y`. Entre ellas introduciremos el contenido que deseamos. Mejor expliquémoslo con un ejemplo tomado de http://www.w3schools.com/js/tryit.asp?filename=tryjs_myfirst:

En este ejemplo hay dos partes importantes:

```
<button type="button"
    onclick="document.getElementById('demo').innerHTML
        = Date()">
    Click me to display Date and Time.</button>
<p id="demo"></p>
```

Estas partes definirán un botón HTML y qué acción ha de invocar cuando se pulsa en dicho botón. Si vemos el contenido , vemos que contiene el nombre de la función . El tag contiene el destino donde se insertará el resultado de la acción.

Expliquemos por partes el código:

- ▶ : corresponde al elemento que define la página HTML en JavaScript. Es decir, cuando escribimos document es porque pretendemos realizar alguna acción con el contenido HTML.
- ▶
 - Como hemos visto antes, define el destino. Si miramos cómo se definía, veremos que contenía un id, que es el mismo que encontramos entre paréntesis después de . Lo único que estamos diciendo con esta función es que vamos a trabajar con el contenido de ese tag.
 - Además, incluyendo estamos indicando que vamos a trabajar con el contenido HTML.
 - Date es una función definida ya en JavaScript que nos da la hora y la fecha en

el momento en que se invoca.

Es decir, lo que se muestra en este ejemplo es un botón que, al pulsarlo, nos introducirá la fecha y hora actual en la página mostrada por el navegador.

A continuación explicaremos Brackets, donde podremos experimentar con JavaScript, crear nuevos archivos y conectarlos.

1.4. Editor código: Brackets

Brackets es un editor de código fuente abierto, gratuito y creado por la empresa Adobe Systems, orientado al desarrollo web. Está enfocado hacia el desarrollo web de *frontend* (HTML, CSS y JavaScript).

Accede a la página oficial de Brackets:

<http://brackets.io/>

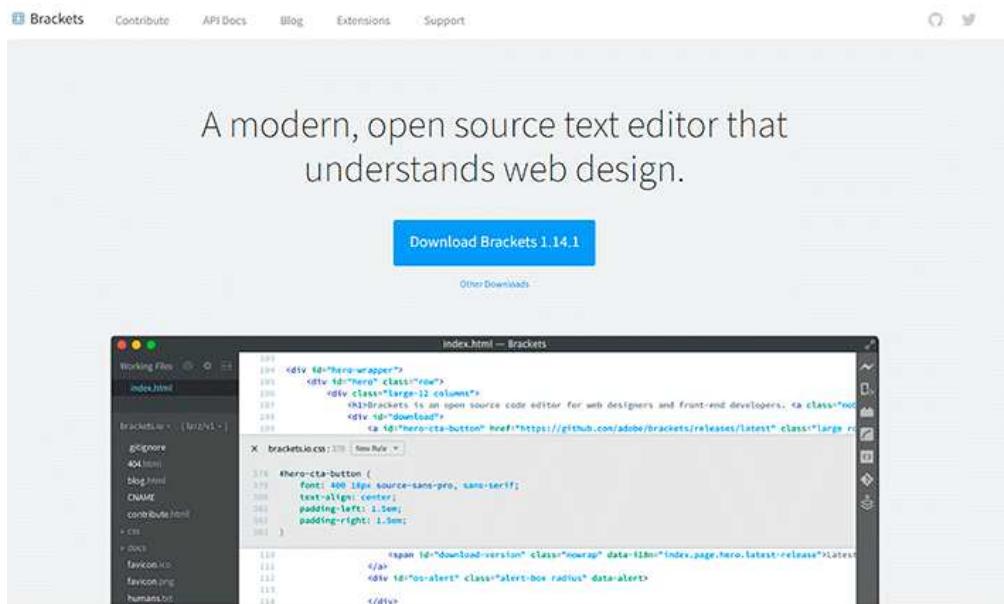


Figura 5. Brackets. Fuente: <http://brackets.io/>

Se trata de una herramienta *open source* con una interfaz sencilla y minimalista, que incorpora ayudas en la escritura del código y sangrado en las sentencias de código, lo cual permite que este sea mucho más legible, todo ello con una organización de todos los elementos del proyecto web con base en proyectos.

Otra facilidad incorporada en Brackets es que utiliza un código de colores bastante visible. Es decir, cuando haces clic en cualquier etiqueta HTML, se marca en color verde tanto el principio como el fin del párrafo, siempre que esté bien cerrado. Si no es así, el resto de etiquetas posteriores las marca en rojo. Esta facilidad ayuda a no cometer errores debido a etiquetas que queden abiertas.

Una de sus características más importantes es el **modo de vista previa en vivo**. En este modo podemos abrir el proyecto web en el navegador y ver en tiempo real las modificaciones que realicemos en el código fuente HTML + CSS + JavaScript. Es una característica enormemente útil porque nos permite modificar cualquier elemento del código y ver reflejados dichos cambios de forma instantánea en la ventana de previsualización, gracias a que incorpora un servidor web local.

Brackets también incorpora la edición rápida de etiquetas. Es decir, si hacemos clic en etiquetas como h1, p, etc., y luego hacemos «Ctrl + E», se despliega el estilo CSS que tenga asociado. Además, nos permite editar dicho CSS desde este punto sin tener que acudir al archivo CSS.

En definitiva, Brackets es una muy buena opción tanto si quieres aprender editando código nuevo como para revisar el código de los ejercicios hechos en clase.

Si necesitas profundizar en este editor, aquí tienes la guía oficial:

<https://github.com/adobe/brackets/wiki/How-to-Use-Brackets>

1.5. Referencias bibliográficas

Brackets: A modern, open source code editor. <http://brackets.io/>

Carto. <https://carto.com/>

D3.js: Data-Driven Documents. <https://d3js.org/>

Gartner. (11 de febrero de 2020). *Gartner Magic Quadrant for Analytics and Business Intelligence Platforms.* <https://www.gartner.com/en/documents/3980852/magic-quadrant-for-analytics-and-business-intelligence-p>

Google Developers. (S. f.). *Google Charts.* <https://developers.google.com/chart>

Microsoft. (S. f.). *Power BI: Microsoft Power Platform.* <https://powerbi.microsoft.com/>

Qlik: Analítica de datos eficaz. <https://www.qlik.com/>

The R Foundation. (S. f.). *The R Project for Statistical Computing.* <https://www.r-project.org/>

Tableau: Software de análisis e inteligencia de datos. <https://www.tableau.com/>

The 27 Worst Charts of All Time

Hickey, W. (26 de junio de 2013). The 27 Worst Charts of All Time. *Business Insider*.

<http://www.businessinsider.com/the-27-worst-charts-of-all-time-2013-6?op=1>

Este artículo nos muestra una serie de gráficos que, aunque innovadores, fueron un fracaso.

Graphing guidelines

Mr. Guch. (18 de marzo de 2012). Graphing guidelines. *Mrphysics*.
<http://mrphysics.org/MrGuch/graph.html>

En este artículo encontrarás algunas directrices para realizar un buen gráfico.

Pie charts: the bad, the worst and the ugly

Frey, L. (5 de febrero de 2013). Pie charts: the bad, the worst and the ugly. *Visuanalyze*. http://visuanalyze.wordpress.com/2013/02/05/bad_piecharts/

En este artículo puedes ver por qué un gráfico circular, uno de los favoritos para la visualización de datos, no siempre es adecuado.

- 1.** Para explicar una historia basada en datos debemos:
 - A. Entender los datos disponibles.
 - B. Ser capaz de representarlos gráficamente con las herramientas adecuadas.
 - C. Comprobar que los destinatarios entiendan el mismo mensaje que pretendes explicar.
 - D. Todas las anteriores.

- 2.** Los datos y sus relaciones no son siempre algo fácil de entender:
 - A. Correcto y por ello las herramientas de visualización nos pueden ayudar a tratar con información compleja.
 - B. Correcto porque los datos siempre requieren simplificar su complejidad.
 - C. Incorrecto si estamos tratando con datos de carácter personal.
 - D. Incorrecto si estamos tratando con datos numéricos.

- 3.** Todas las herramientas de visualización requieren de conocimientos de programación:
 - A. Incorrecto.
 - B. Correcto. Tableau es un ejemplo de ello.
 - C. Correcto. Google Chart es un ejemplo de ello.
 - D. Correcto. D3.js y Carto son un ejemplo de ello.

- 4.** ¿Cuáles de estas herramientas están orientadas a usuarios no desarrolladores?
 - A. Tableau.
 - B. Qlik.
 - C. Carto.
 - D. Todas las anteriores.

5. ¿Cuáles de estas herramientas están orientadas a usuarios desarrolladores?
 - A. Google Chart.
 - B. Carto.
 - C. Qlik.
 - D. Todas las anteriores.

6. En la actualidad las principales plataformas modernas de analítica e inteligencia de negocio...
 - A. Se diferencian especialmente por sus capacidades en la visualización de datos.
 - B. Ya no se diferencian especialmente por sus capacidades en la visualización de datos.
 - C. Requieren de usuarios avanzados en programación.
 - D. Solo son accesibles por empresas.

7. Enumera, según Gartner, 3 características de las herramientas modernas de visualización de datos la actualidad:
 - A. Análisis avanzado, *insights* automatizados y catálogo automático.
 - B. Seguridad, *cloud* y conectividad a fuentes de datos.
 - C. Seguridad, soporte a la preparación de los datos y consulta mediante búsquedas en lenguaje natural.
 - D. Todas son correctas.

8. Una página web se construye con base en los siguientes niveles:
 - A. HTML y CSS.
 - B. HTML y editores de código.
 - C. Captura de datos y visualización.
 - D. Estructura, contenido, apariencia y comportamiento.

- 9.** Se dice que HTML es un lenguaje de marcas porque:
- A. Permite las llamadas a APIs o servicios externos de terceros.
 - B. Señaliza todas las líneas de código haciéndolas más legibles.
 - C. Es compatible con otros lenguajes como JavaScript.
 - D. Codifica con base en etiquetas.
- 10.** Brackets es un editor de código:
- A. Compatible con HTML, CSS y JavaScript
 - B. Que incorpora un servidor web local.
 - C. Que permite ver en tiempo real las modificaciones que realicemos en el código fuente.
 - D. Todas son correctas.

Herramientas de Visualización

Tema 2. Google Chart. Introducción y principales visualizaciones

Índice

[Esquema](#)

[Ideas clave](#)

[2.1. Introducción y objetivos](#)

[2.2. Ejemplos de varias visualizaciones](#)

[2.3. Conectando con Google Spreadsheets y archivos CSV](#)

[2.4. Gestionar eventos](#)

[2.5. Referencias bibliográficas](#)

[A fondo](#)

[Controls and Dashboards](#)

[Ejemplos de Google Chart](#)

[Google Chart eventos](#)

[Test](#)



2.1. Introducción y objetivos

Google Chart Library es la librería de visualizaciones utilizada por Google Spreadsheets. Google utiliza esta funcionalidad en la hoja de cálculo que han implementado y esto nos permite infinitas posibilidades donde estas librerías son una pequeña funcionalidad.

Piensa en *real-time analytics*, potentes algoritmos procesando los datos en el *backend*, visualizaciones de la actividad en Twitter con los contenidos de los *tweets* en sí..., un gran abanico de posibilidades donde, como expertos en *analytics*, pondréis el límite a las aplicaciones de la empresa con la que colabores. Al fin y al cabo, las visualizaciones son la representación alternativa de la información.

Empezaremos con una simple frase sobre Google y sus funcionalidades, entre las que incluiremos Google Chart Library:

«Google has the functionality of a really complicated Swiss Army knife, but the home page is our way of approaching it closed. It's simple, it's elegant. You can slip it in your pocket, but it's got the great doodad when you need it». (Tischler, 2005).

A partir de aquí empiezan las *hand-on sessions*, practicar y programar será nuestra premisa. Si encuentras dificultades en seguir JavaScript, no dudes en volver al tema anterior y dedicar un poco de tiempo al enlace que incluye el tutorial de JavaScript. No nos asustemos, siguiendo y repitiendo los pasos que indicamos a lo largo del tema, se puede aprender fácilmente cómo trabajar con las visualizaciones.

Nos centraremos en los siguientes aspectos:

- ▶ Ejemplos de **visualizaciones**: aprenderemos a través de ejemplos existentes en Internet.
- ▶ Conectar con **datos externos**: para que no tengamos que modificar el código cada vez que queramos cambiar los valores.

- ▶ **Eventos:** para poder conectar nuestras visualizaciones con otras herramientas.

2.2. Ejemplos de varias visualizaciones

Pie chart

Empezaremos por un *pie chart* y con un ejemplo muy fácil que podemos encontrar en Quickstart de Google:

https://developers.google.com/chart/interactive/docs/quick_start

Ejemplo: código completo de un *pie chart*

```
<html>
<head>
<script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js">
</script>
<script type="text/javascript">
// Load Charts and the corechart package.
google.charts.load('current', {'packages':['corechart']});
//Draw the pie chart for Sarah's pizza when Charts is
loaded.
google.charts.setOnLoadCallback(drawChart);
// Callback that draws the pie chart for Sarah's pizza.
function drawChart() {
// Create the data table for Sarah's pizza.
var data = new google.visualization.DataTable();
data.addColumn('string', 'Topping');
data.addColumn('number', 'Slices');
data.addRows([
['Mushrooms', 1],
['Onions', 1],
['Olives', 2],
['Zucchini', 2],
['Pepperoni', 1]
]);
// Set options for Sarah's pie chart.
var options = {title:'How Much Pizza Sarah Ate Last
Night',
width:400,
height:300};
```

```
// Instantiate and draw the chart for Sarah's pizza.  
var chart = new  
google.visualization.PieChart(document.getElementById  
('Sarah_chart_div'));  
chart.draw(data, options);  
}  
</script>  
</head>  
<body>  
<!--Table and divs that hold the pie charts-->  
<table class="columns">  
<tr>  
<td>  
<div id="Sarah_chart_div" style="border: 1px solid #ccc">  
</div></td>  
</tr>  
</table>  
</body>  
</html>
```

Este código lo puedes copiar en un fichero de texto, guardarlo con la terminación «.html» y abrirlo con un navegador o con el editor de código Brackets que vimos en el tema anterior. Vamos a separarlo por partes para entenderlo mejor.

Parte exclusiva de HTML

```
< html>  
<head>  
</head>  
<body>  
<!--Table and divs that hold the pie charts-->  
<table class="columns">  
<tr>  
<td><div id="Sarah_chart_div" style="border: 1px solid #ccc">  
</div></td>  
</tr>  
</table>  
</body>  
</html>
```

Esta es la parte exclusiva de HTML. Aquí te tienes que fijar en que hemos creado una sección dentro del HTML que tiene el identificador . Aquí es donde cargaremos nuestra gráfica.

¿Cómo cargamos la librería de Google Chart a nuestro HTML?

```
<script src="https://www.gstatic.com/charts/loader.js"></script>
<script>
google.charts.load('current', {packages: ['corechart']});
google.charts.setOnLoadCallback(drawChart);
...
</script>
```

La primera línea de este ejemplo se ocupa de cargar el propio «cargador» o de la librería. Esta sentencia solo debe ejecutarse una vez, independientemente de cuántas gráficas vayamos a pintar.

Lo que sí debemos hacer es cargar aquella librería asociada a las gráficas que queramos utilizar. En nuestro ejemplo cargamos la librería , que incluye los paquetes de gráficas tipo *bar*, *column*, *line*, *area*, *stepped area*, *bubble*, *pie*, *donut*, *combo*, *candlestick*, *histogram* y *scatter*. Los diferentes tipos de librerías y las gráficas que incluyen se encuentran en la sección Loading de cada tipo de gráfica.

Puedes acceder a la galería de gráficas a través del siguiente enlace:

<https://developers.google.com/chart/interactive/docs/gallery>

Podemos fijarnos en el detalle de que *packages* es plural y está definido con un *array*, por lo que podríamos cargar más de un paquete a la vez. Ejemplo:

```
google.charts.load('current', {packages:
['corechart', 'table', 'sankey']});
```

Por último, tenemos la siguiente línea que llama a la función solo cuando se termine de cargar los paquetes de la librería que hemos mencionado anteriormente.

JavaScript se ejecuta de forma secuencial y esta línea evita que se ejecute la función que pinta la gráfica sin estar cargada la librería, lo que daría un error. Además, en ese momento, el HTML está ya completamente cargado en el explorador.

```
// Draw the pie chart for Sarah's pizza when Charts is loaded.  
google.charts.setOnLoadCallback(drawChart);
```

¿Qué es lo que hace nuestra función drawChart?

Primero hemos de crear nuestros datos.

```
// Create the data table for Sarah's pizza.  
var data = new google.visualization.DataTable();  
data.addColumn('string', 'Topping');  
data.addColumn('number', 'Slices');  
data.addRows([  
  ['Mushrooms', 1],  
  ['Onions', 1],  
  ['Olives', 2],  
  ['Zucchini', 2],  
  ['Pepperoni', 1]  
]);
```

es una estructura de datos con la que trabaja JavaScript, no muy diferente a lo que podría ser un *array de arrays* en Java o C.

Añadimos dos columnas asignándoles un nombre de cabecera que la librería puede utilizar en caso de que sea necesario. Además, indicamos el tipo de valores que contienen las celdas.

Finalmente, añadimos las filas con los valores que contengan. En este caso, añadimos todas las filas de una vez, pero también podríamos añadir las filas una por una. Esto puede ser útil si queremos hacer nuestra **visualización dinámica**, es decir, que cambie basándose en eventos.

```
data.addRow(['Pepperoni', 2]);
```

Después se definen las opciones del gráfico como **título y dimensiones**:

```
// Set options for Sarah's pie chart.  
var options = {title:'How Much Pizza Sarah Ate Last Night',  
width:400,  
height:300};
```

Aquí debemos mirar las opciones que cada *chart* nos ofrece. Por ejemplo, los *pie charts* ofrecen más opciones. Probad estas dos variantes:

```
var options = {  
width: 400,  
height: 240,  
title: 'Toppings I Like On My Pizza',  
colors: ['#e0440e', '#e6693e', '#ec8f6e', '#f3b49f', '#f6c7b6']  
};
```

Y el otro ejemplo:

```
var options = {  
width: 400,  
height: 240,  
title: 'Toppings I Like On My Pizza',  
colors: ['#e0440e', '#e6693e', '#ec8f6e', '#f3b49f', '#f6c7b6'],  
is3D: true  
};
```

Estos dos ejemplos modifican los colores y las dimensiones. Recuerda la teoría de visualizaciones antes de aplicar efectos y colores; siempre es más atractivo a la vista

aplicar efectos 3D y usar diferentes colores o gradientes de colores, sin embargo, puede jugar en contra de vuestras visualizaciones. No todo el mundo detecta los mismos gradientes ni percibe de la misma manera las dimensiones.

Finalmente, nos encontramos el código que dibuja nuestro gráfico.

```
// Instantiate and draw the chart for Sarah's pizza.  
var chart = new  
google.visualization.PieChart(document.getElementById  
('Sarah_chart_div'));  
chart.draw(data, options);
```

Instanciamos primero nuestro gráfico, en este caso un *pie chart*. Además, indicamos en la instancia dónde debe ser dibujado.

Cuando iniciamos la acción de dibujar, , le pasamos los valores data que queremos visualizar y las opciones que hemos definido. Con estos sencillos pasos, podemos generar el siguiente gráfico:

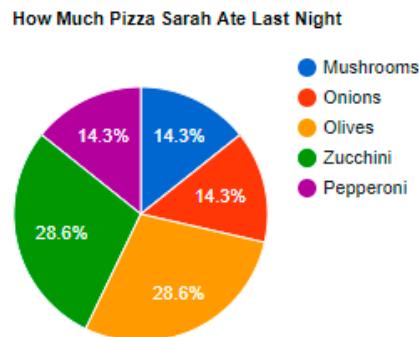


Figura 1. *Pie Chart*. Fuente: https://developers.google.com/chart/interactive/docs/quick_start

A continuación vamos a explicar las demás visualizaciones a partir de las diferencias respecto a la que hemos explicado. El cambio de gráfica es muy sencillo. Basta con especificar el tipo de gráfica deseada.

Horizontal bar chart

Especificamos en el código el tipo de gráfica :

```
var chart = new  
google.visualization.BarChart(document.getElementById  
('Sarah_chart_div'));
```

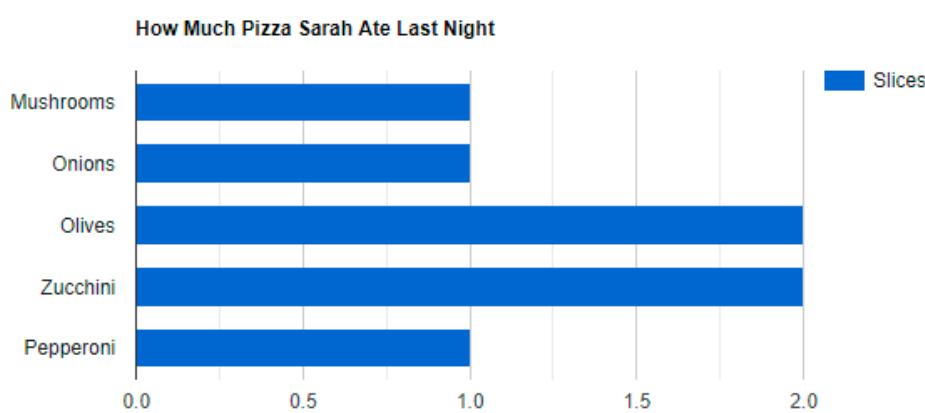


Figura 2. Horizontal bar chart.

Column bar chart

Especificamos en el código el tipo de gráfica :

```
var chart = new  
google.visualization.ColumnChart(document.getElementById  
('Sarah_chart_div'));
```

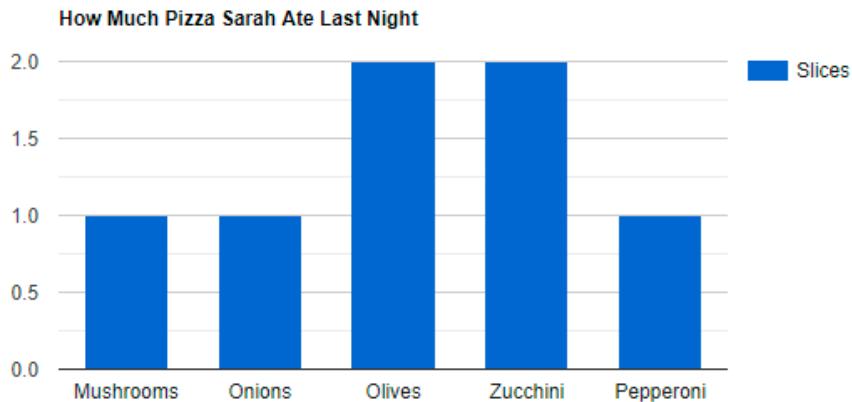


Figura 3. Column bar chart.

Geo chart

Cambiamos completamente de tipo de visualización, pero será más fácil si ya se domina las anteriores gráficas. La librería permite para mostrar un país, un continente o una región.

En lugar de cargar la librería , cargamos la librería .

```
google.charts.load('current', { 'packages': ['geochart'] });
```

Cambiamos nuestra estructura de datos para **definir localizaciones**:

```
var data = google.visualization.arrayToDataTable([
['Country', 'Popularity'],
['Germany', 200],
['United States', 300],
['Brazil', 400],
['Canada', 500],
['France', 600],
['RU', 700]
]);
```

Luego dibuja el mapa:

```
var chart = new google.visualization.GeoChart  
(document.getElementById('regions_div'));  
chart.draw(data, options);
```

Este es el resultado que deberías tener:

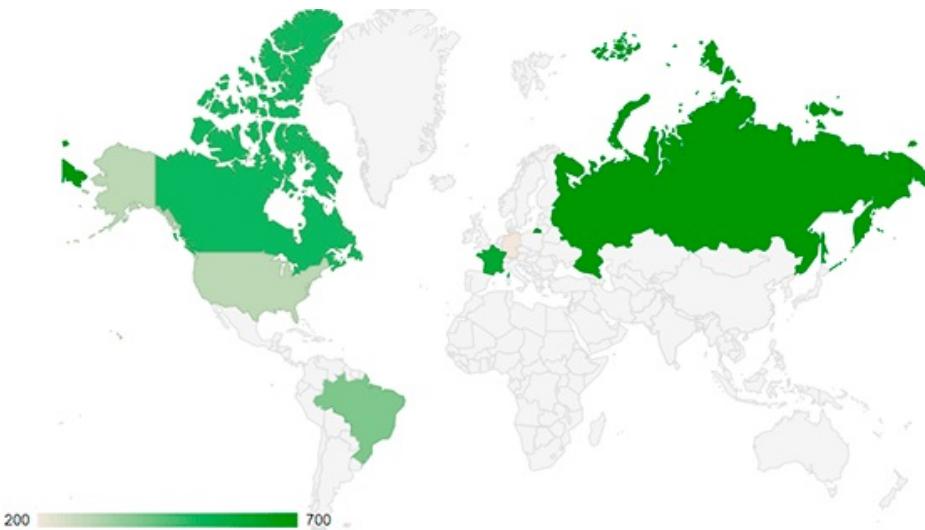


Figura 4. *Geo chart.*

Intenta modificar el código anterior.

Puedes acceder al código completo a través del siguiente enlace:

<https://developers.google.com/chart/interactive/docs/gallery/geochart>

2.3. Conectando con Google Spreadsheets y archivos CSV

Los datos de las visualizaciones no tienen por qué estar incrustados en el código fuente de nuestro fichero .html. Si fuera así, cada vez que quisieramos mostrar información, deberíamos abrirlo y modificarlo, lo cual no es ni práctico ni realista. Para ello las visualizaciones recurren a leer los datos de **fuentes de datos externas**.

Google Spreadsheets

Si queremos evitar modificar el código JavaScript cada vez que modificamos los datos, podemos conectar los gráficos a hojas de cálculo existentes. El proceso se explica en el siguiente enlace.

Google Spreadsheets:

<https://developers.google.com/chart/interactive/docs/spreadsheets>

Pero intentaremos extender y explicar por partes el código. El código completo es el siguiente:

Ejemplo: código completo de Google Spreadsheets

```
<html>
<head>
<script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
<script type="text/javascript">
google.charts.load("current", {packages: ["corechart"]});
google.setOnLoadCallback(drawChart);
function drawChart() {
var query = new google.visualization.Query(
'https://docs.google.com/spreadsheet/ccc?
key=0Atw2BTU52l0CdEZpUlVIdmxG0WZBR2tuLXhYN2dQTWc&usp
=drive_web#gid=0');
query.send(handleQueryResponse);
```

```
}

function handleQueryResponse(response) {
if (response.isError()) {
alert('Error in query: ' + response.getMessage() + ' ' +
response.getDetailedMessage());
return;
}
var data = response.getDataTable();
var chart = new google.visualization.
ColumnChart(document.getElementById('columnchart'));
chart.draw(data, { legend: { position: 'none' } });
}
</script>
<title>Data from a Spreadsheet</title>
</head>
<body>
<span id='columnchart'></span>
</body>
</html>
```

Te llamará la atención que la función `handleQueryResponse` ha cambiado:

```
function drawChart() {
var query = new google.visualization.Query(
'https://docs.google.com/spreadsheet/ccc?
key=0Atw2BTU52l0CdEZpUlVIdmxG0WZBR2tuLXhYN2dQTWc&usp=drive_web#gid=0');

query.send(handleQueryResponse);
```

Para utilizar Google Spreadsheet como fuente de datos se necesita:

- ▶ Asignar a la hoja de datos permisos de lectura Público en la web o Cualquiera con el enlace.
- ▶ Disponer de la URL completa del documento.
- ▶ Llamar a la función `handleQueryResponse` con la URL.
- ▶ Especificar los parámetros y .

- indica cuantas filas son cabecera y por lo tanto deben excluirse como datos.
- se refiere a la hoja a utilizar (en el caso de que el documento tuviera varias hojas).

Puedes copiar la URL y pegarla en tu explorador para ver el contenido.

	A	B
1	Name	Length
2	A	23
3	B	16
4	C	10
5	D	31
6		
7		

Figura 5. Ejemplo de Google Spreadsheet.

El resultado que se obtiene se envía a otra función llamada , cuya función es doble:

Gestionar el error en la petición del documento. ¿Qué errores consideramos en este caso? Por ejemplo, que la URL del documento no sea correcta. Probadlo en vuestro navegador o en Brackets, borrad uno de los caracteres de la URL y veremos cómo aparece una ventana reportando un error.

Dibujar la gráfica con el método . Para ello se siguen los siguientes pasos:

- ▶ Asignamos los datos al gráfico a través de la variable response. El objeto response se pasa a la función cuando se llama a la función .
- ▶ Las opciones de la gráfica se definen al llamar a la función .

La **función** dibuja la visualización de la gráfica y tiene dos parámetros:

- ▶ : alberga los datos a usar para dibujar la gráfica.
- ▶ : mapa con duplas de valores en formato nombre y valor. Ejemplo:

```
{x:100, y:200, title:'An Example'}  
function handleQueryResponse(response) {  
if (response.isError()) {  
alert('Error in query: ' + response.getMessage() + ' ' +  
response.getDetailedMessage());  
return;  
}  
var data = response.getDataTable();  
var chart = new google.visualization.  
ColumnChart(document.getElementById('columnchart'));  
chart.draw(data, { legend: { position: 'none' } });  
}
```

El resultado de este código es el siguiente:

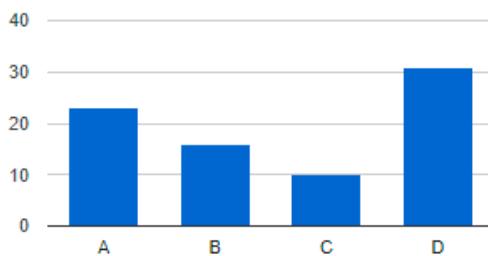


Figura 6. Resultado visualización datos Google spreadsheet.

Accede más información sobre la función y las opciones a utilizar a través del siguiente enlace:

<https://developers.google.com/chart/interactive/docs/reference#visdraw>

Archivos CSV

Los archivos CSV (*comma-separated values* o valores separados por comas) son un tipo de documento de texto ampliamente utilizado para **representar datos en forma de tabla**. Su característica principal es que las columnas se separan por comas y las filas por saltos de línea.

Los archivos CSV normalmente se utilizan para importar o exportar información de bases de datos.

En este ejercicio también utilizaremos jQuery, una librería de código abierto de JavaScript, para el tratamiento del fichero CSV. El código adaptado sería el siguiente:

Ejemplo: código adaptado de CSV

```
<html>
<head>
<title>Google Chart Example</title>
<script type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/
jquery.min.js">
</script>
<script src="jquery.csv-0.71.js"></script>
<script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
<script type="text/javascript">
google.charts.load("current", {packages: ["corechart"]});
google.setOnLoadCallback(drawChart);

function drawChart() {
// grab the CSV
$.get("https://docs.google.com/spreadsheets/pub?
key=0AhNdznUnSazTdElCVTRTYzF6Zm9DSmlUQkx1NE5DY1E&output
=csv",
function(csvString) {
// Transformar CSV a array utilizando jQuery
var arrayData = $.csv.toArrays(csvString, {onParseValue:
$.csv.hooks.castToScalar});
// Convertir array a DataTable
var data =
new google.visualization.arrayToDataTable(arrayData);
// Generar gráfica desde el DataTable
var chart = new
google.visualization.ColumnChart(document.getElementById
```

```
('columnchart'));
chart.draw(data, { legend: { position: 'none' } });
});
}
</script>
</head>
<body>
<div id="columnchart"></div>
</body>
</html>
```

Accede al fichero CSV que utilizaremos en el ejemplo a través del siguiente enlace:

<https://docs.google.com/spreadsheet/pub?key=0AhNdznUnSazTdElCVRTYzF6Zm9>

Respecto al anterior ejemplo, resaltaremos el siguiente código:

```
$.get("https://docs.google.com/spreadsheet/pub?
key=0AhNdznUnSazTdElCVRT
YzF6Zm9DSmlUQkx1NE5DY1E&output=csv", function(csvString) {
// transform the CSV string into a 2-dimensional array
var arrayData = $.csv.toArrays(csvString, {onParseValue:
$.csv.hooks.castToScalar});
...
});
```

Lo más importante a tener en cuenta es que el `$.get` es un **formato** utilizado en jQuery.

Esto crea una función y todo el código que va a continuación está contenido en dicha función:

- ▶ En cierta manera, cuando vemos el símbolo `$` es equivalente al `document` de JavaScript estándar. Quiere decir que vamos a hacer algo con el documento HTML.
- ▶ El `$.get` realiza una petición http a la URL indicada.

A continuación es cuando la librería de jQuery para ficheros CSV nos ayuda, haciendo una llamada a `$.csv.toArrays`, transformamos el fichero CSV a un formato que es

comprendible para la librería de Google Charts.

El resultado es el mismo que el obtenido desde la fuente de Google Spreadsheets, pero en este caso, tomando como referencia un fichero en formato CSV.

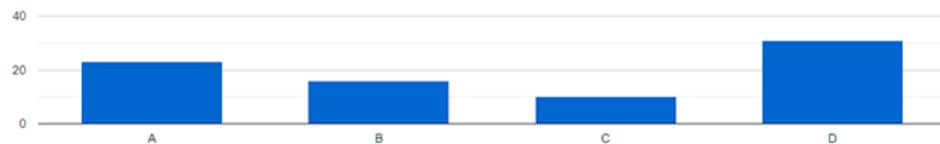


Figura 7. Resultado visualización datos del fichero CSV.

2.4. Gestionar eventos

Google Charts genera eventos que se pueden tratar y gestionar. Estos eventos se suelen disparar por acciones de usuario, como por ejemplo cuando se hace clic en una gráfica. Se puede registrar un método JavaScript que sea llamado cuando ocurran dichos eventos e incluso acceder a la información asociada a los mismos.

Cada gráfica y cada interacción definen sus propios eventos y la información asociada. En la documentación de Google Chart explican con detalle los posibles eventos:

Accede a más información sobre eventos a través del siguiente enlace:

<https://developers.google.com/chart/interactive/docs/events>

En esta sección explicaremos de forma sencilla cómo podemos trabajar con los eventos para **interactuar entre dos visualizaciones**.

Ejemplo: código de dos visualizaciones

```
<html>
<head>
<script type="text/javascript"
src="https://www.google.com/jsapi"></script>
<script type="text/javascript">
google.load('visualization', '1.0',
{'packages':['corechart','table']});
// Set a callback to run when the Google
Visualization API is loaded.
google.setOnLoadCallback(drawSort);
function drawSort() {
var data = new google.visualization.DataTable();
data.addColumn('string', 'Name');
data.addColumn('number', 'Salary');
data.addColumn('boolean', 'Full Time');
data.addRows(5);
data.setCell(0, 0, 'John');
```

```
data.setCell(0, 1, 10000);
data.setCell(0, 2, true);
data.setCell(1, 0, 'Mary');
data.setCell(1, 1, 25000);
data.setCell(1, 2, true);
data.setCell(2, 0, 'Steve');
data.setCell(2, 1, 8000);
data.setCell(2, 2, false);
data.setCell(3, 0, 'Ellen');
data.setCell(3, 1, 20000);
data.setCell(3, 2, true);
data.setCell(4, 0, 'Mike');
data.setCell(4, 1, 12000);
data.setCell(4, 2, false);
var formatter = new google.visualization.NumberFormat
({prefix: '$'});
formatter.format(data, 1); // 
    Apply formatter to second column
var view = new google.visualization.DataView(data);
view.setColumns([0, 1]);
var table = new
google.visualization.Table(document.getElementById
('table_sort_div'));
table.draw(view);
var chart = new
google.visualization.BarChart(document.getElementById
('chart_sort_div'));
chart.draw(view);
google.visualization.events.addListener(table, 'sort',
function(event) {
data.sort([{column: event.column, desc:
!event.ascending}]);
chart.draw(view);
}
);
}
</script>
</head>
<body>
<!--Div that will hold the pie chart--&gt;
&lt;div id="chart_sort_div"&gt;&lt;/div&gt;
&lt;div id="table_sort_div"&gt;&lt;/div&gt;</pre>
```

```
</body>  
</html>
```

En este último ejemplo de código prestad atención a dos detalles, ya que tenemos:

- ▶ Dos visualizaciones: una tabla y un *bar chart*.
- ▶ La función :

```
var table = new  
google.visualization.Table(document.getElementById('table_sort_div'));  
  
table.draw(view);  
var chart = new  
google.visualization.BarChart(document.getElementById('chart_sort_div'));  
  
chart.draw(view);  
google.visualization.events.addListener(table, 'sort',  
function(event) {  
data.sort([{column: event.column, desc: !event.ascending}]);  
chart.draw(view);  
});
```

La función `es la que utilizaremos si queremos trabajar con eventos. Lo que creamos con esta función es una entidad que «escuchará» los eventos de las visualizaciones.`

En este código creamos un que escucha los eventos de nuestro objeto `table` (primera variable de la función). Escucha solo un tipo de eventos: Es decir, cuando un usuario ordena la tabla que normalmente esa acción genera al pulsar en la cabecera de la tabla, el evento se lanza. Una vez que el escucha el evento, ordena nuestro objeto `data` con la función y redibuja el *bar chart*.

El resultado de este código es el siguiente: en la gráfica superior los datos están ordenados de mayor a menor salario. En la gráfica inferior, su visualización ha

cambiado automáticamente cuando hemos ordenado los datos de la tabla en sentido inverso, de menor a mayor.

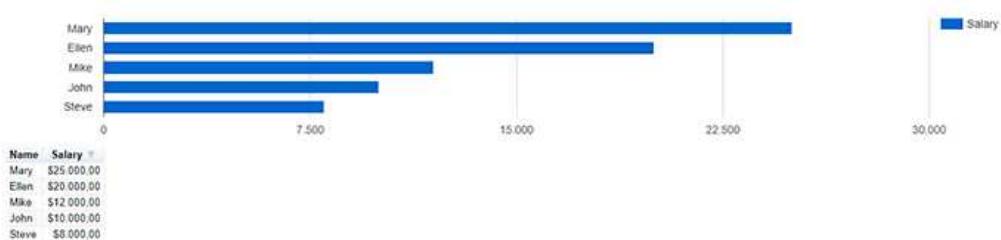


Figura 8. Resultado visualización datos ordenados de mayor a menor salario.

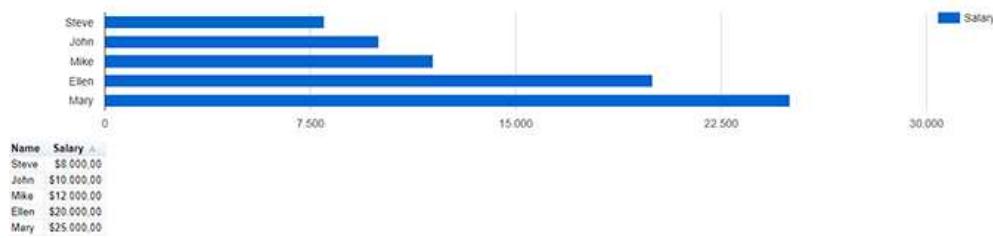


Figura 9. Resultado visualización datos ordenados de menor a mayor salario.

2.5. Referencias bibliográficas

Tischler, L. (1 de noviembre de 2005). The Beauty of Simplicity. *Fast Company*.

<https://www.fastcompany.com/56804/beauty-simplicity>

Controls and Dashboards

Google Developers. *Controls and Dashboards.* Google Charts.

<https://developers.google.com/chart/interactive/docs/gallery/controls>

Esta guía muestra cómo construir un *dashboard* con Google Charts.

Ejemplos de Google Chart

Google

Developers. *Chart*

Gallery.

Google

Charts.

<https://developers.google.com/chart/interactive/docs/gallery>

En Chart Gallery se pueden ver y probar más ejemplos de Google Charts.

Google Chart eventos

Google

Developers. *Code*

Examples.

Google

Charts.

<https://developers.google.com/chart/interactive/docs/examples>

En Code Examples se puede ver y gestionar eventos de Google Charts.

1. necesita ser instanciada con un new para crear un objeto:

 - A. Sí.
 - B. No, solo cuando le asignas valores directamente.
 - C. No, solo cuando trabajas la primera vez con la gráfica.
 - D. No.

2. ¿Qué función tiene ?

 - A. Carga el propio cargador de librerías.
 - B. Importa la librería *jQuery*.
 - C. Carga los diferentes paquetes de la librería.
 - D. Carga la última versión de Google Charts.

3. La primera variable que recibe la función de JQuery es:

 - A. La URL a la que se dirige la petición.
 - B. El evento que ha generado la visualización.
 - C. El nombre de función destino donde se pasan los valores.
 - D. El *array* de datos a visualizar.

4. añade una columna a nuestra tabla de datos:

 - A. Sí, tantas veces como aparezca esta línea de código.
 - B. Sí, aunque no admite valores iguales.
 - C. No, porque la tabla ya viene con las columnas definidas.
 - D. No.

5. Existen librerías en JQuery que nos facilitan la transformación de archivos CSV a datos comprensibles por la librería de Google Charts:
- A. Sí.
 - B. No, porque jQuery no incorpora dichas funciones.
 - C. No, porque CSV no es un formato compatible con Google Chart.
 - D. Sí, aunque depende de que las versiones sean compatibles.
6. nos dibujará *bar charts* verticales:
- A. Sí, admite tanto barras verticales como horizontales.
 - B. Sí, hace lo mismo que *colum chart*.
 - C. Sí, tanto barras independientes como apiladas.
 - D. No.
7. ¿Qué hace el código cuando contiene ?
- A. Comprueba si el evento generado por otra visualización tiene el formato correcto.
 - B. Comprueba si ha habido algún error con la petición http.
 - C. Comprueba si ha habido algún problema en la generación de la visualización.
 - D. Comprueba si los datos del archivo son coherentes.
8. se utiliza para escuchar eventos que generan las visualizaciones:
- A. Sí.
 - B. No, solo aplica los eventos del ratón.
 - C. No, se encarga de los eventos del sistema operativo.
 - D. No.

- 9.** Todas las visualizaciones de Google Chart generan los mismos tipos de eventos:
- A. Sí, porque todas funcionan de la misma forma.
 - B. Sí, porque de esta forma el código es escalable.
 - C. Sí, porque ello permite cambiar la gráfica sin tener que cambiar los eventos.
 - D. No.
- 10.** ¿Incluiremos el paquete para visualizar un mapa?
- A. Sí, para mostrar un país, un continente o una región.
 - B. Sí, para mostrar incluso direcciones de calles.
 - C. No, porque no existe dicha librería.
 - D. No.

Herramientas de Visualización

Tema 3. D3.js. Introducción y funcionalidades

Índice

[Esquema](#)

[Ideas clave](#)

[3.1. Introducción y objetivos](#)

[3.2. Definición e instalación](#)

[3.3. Elementos básicos de D3.js. Generando elementos HTML](#)

[3.4. Trabajando con datos reales y elementos en el HTML](#)

[3.5. Ventajas de D3.js](#)

[3.6. Referencias bibliográficas](#)

[A fondo](#)

[D3 for the Impatient: Interactive Graphics for Programmers and Scientists](#)

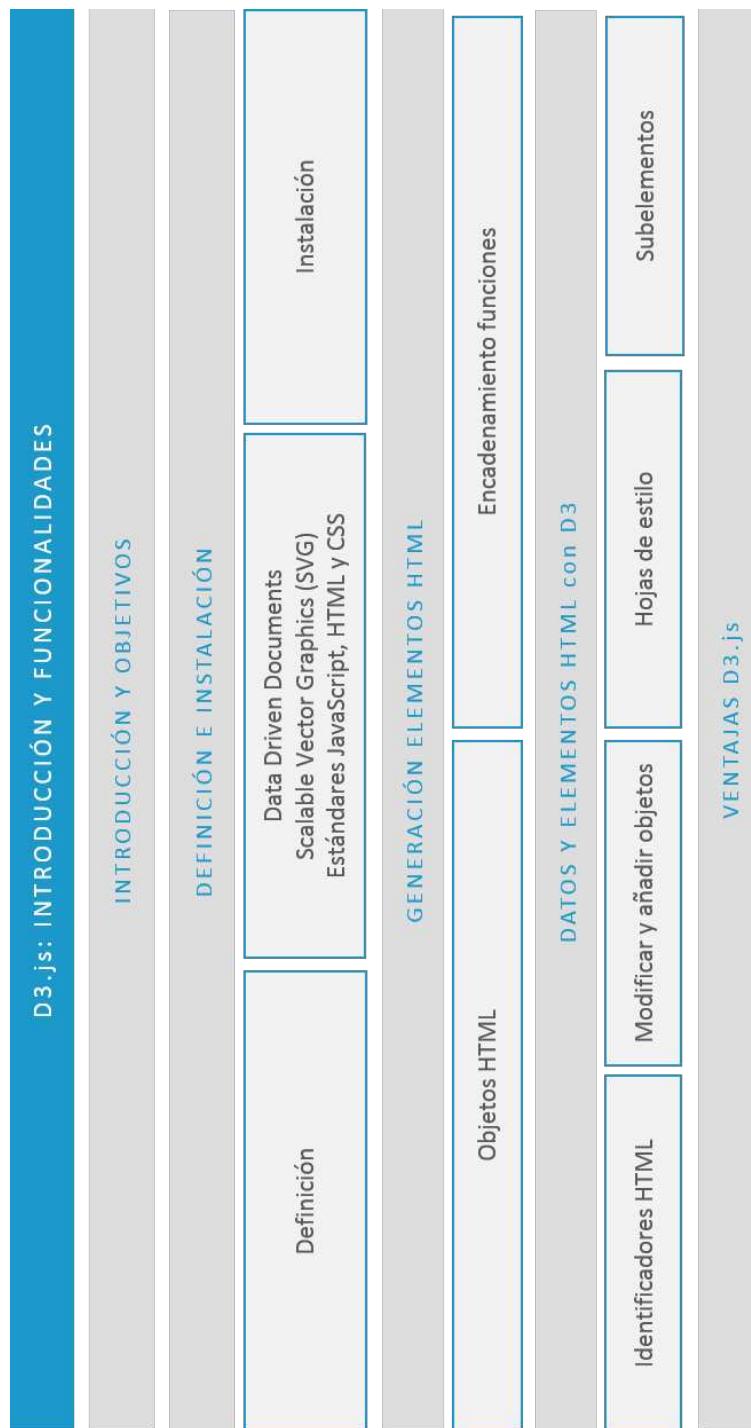
[Data Visualization with D3.js Cookbook](#)

[Data Driven Documents D3.js: Tips and Tricks](#)

[Creando un mapa sencillo con D3](#)

[Learn D3.js in 5 minutes](#)

[Test](#)



3.1. Introducción y objetivos

En los próximos temas nos dedicaremos únicamente a la librería D3. ¿Por qué tantos temas a esta librería, qué tiene de especial? Vamos a conocer a fondo una librería enormemente flexible y que nos permitirá diseñar nuestras gráficas hasta el último detalle. Para ello nos basaremos en JavaScript como lenguaje de programación y abordaremos algunos conceptos avanzados que no suelen aparecer en la programación de páginas en HTML.

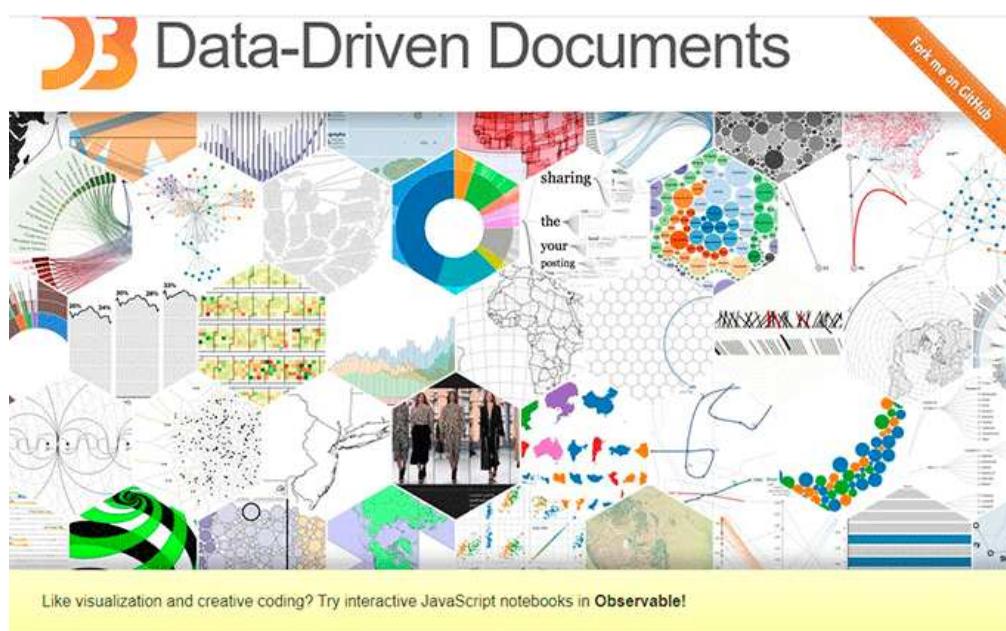


Figura 1. D3.js Data Driven Documents. Fuente: <https://d3js.org/>

Esta librería incluso nos permite crear las visualizaciones, aunque no nos consideremos buenos diseñadores. En GitHub podemos encontrar multitud de ejemplos en los que basarnos y que también podremos modificar.

Accede a la galería de GitHub a través del siguiente enlace:

<https://github.com/d3/d3/wiki/Gallery>

Lo importante es que acabemos este curso siendo capaces de diseñar e implementar nuestras propias visualizaciones, ya sea desde cero o modificando una existente. En concreto este tema introducirá los **conceptos básicos** de D3.js. para ir profundizando en temas posteriores.

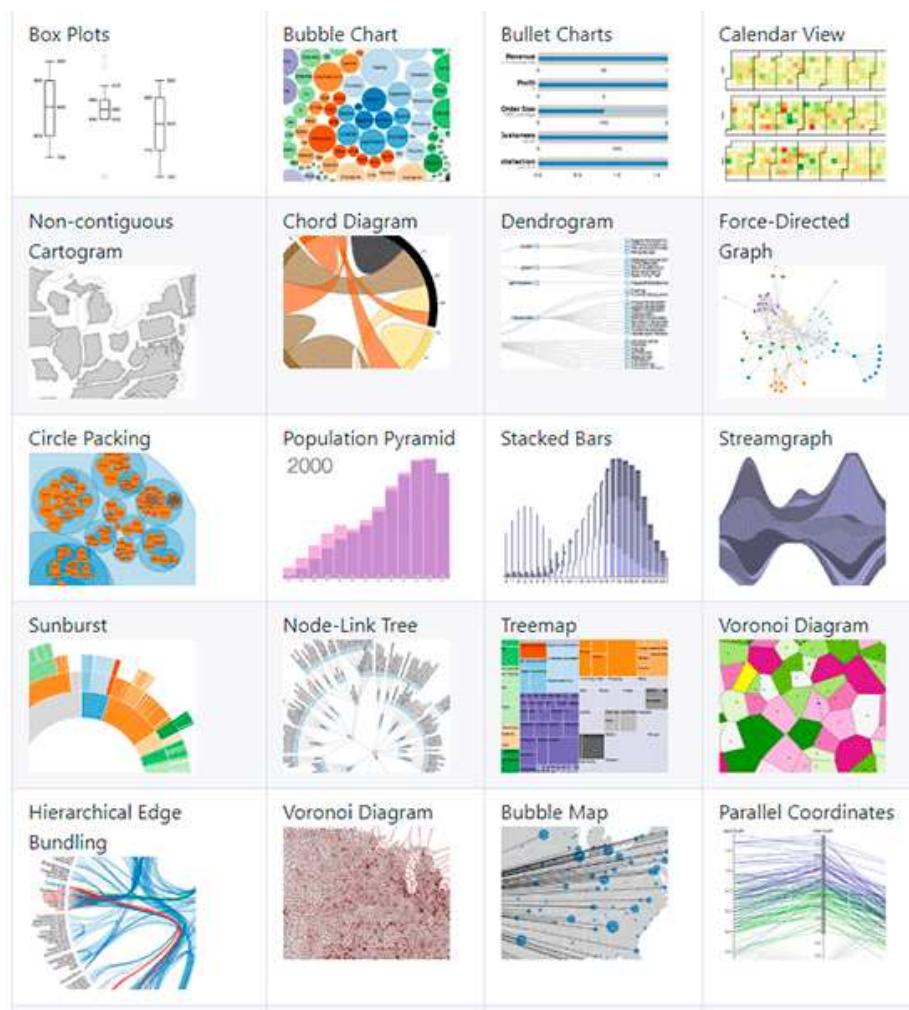


Figura 2. Ejemplos visualizaciones D3. Fuente: <https://github.com/d3/d3/wiki/Gallery>

Comenzamos con una frase de Barbara Sher:

«Puedes aprender cosas nuevas en cualquier momento de tu vida si estás dispuesto a

ser un principiante. Si realmente aprendes a que te guste ser un principiante, el mundo entero se abre ante ti» (100 frases).

Te aconsejamos que en los primeros temas no intentes ir demasiado rápido. Lo importante es que los conceptos queden claros, pues serán muy útiles en visualizaciones más avanzadas y, si no se le presta la atención necesaria, más tarde puedes tener problemas.

Este tema es exclusivamente de introducción, no hay gran complicación tecnológica, pero es necesario para familiarizarnos con D3, por lo que sería interesante ver los recursos que recomendamos en este tema.

3.2. Definición e instalación

Antes de nada, empezaremos definiendo qué es D3.js:

D3.js (*Data-Driven Documents*) es una librería JavaScript que utiliza datos digitales para impulsar la creación y control de gráficas dinámicas e interactivas en los navegadores.

Es una herramienta compatible con W3C, haciendo uso de los estándares en gráficos SVG (*Scalable Vector Graphics*), JavaScript, HTML5 y CSS3 (*Cascading Style Sheets*). A diferencia de otras librerías D3, permite un gran control sobre el resultado visual final.

De esta definición resaltaremos varios conceptos que son relevantes:

Data driven documents

Este concepto es bastante importante si lo que estamos pensando es en **reutilizar visualizaciones existentes**. Hasta ahora, debíamos adaptar nuestra estructura de datos a las visualizaciones, mientras que aquí hay algo más de libertad.

Podemos adaptar nuestros datos o bien podemos cambiar la forma en la que D3 está leyendo los datos y es lo que se denomina **documentos orientados a datos**. D3 se creó con la intención de tener control absoluto sobre la visualización.

Scalable Vector Graphics (SVG).

No importa dónde mostremos nuestra visualización: pantallas y formatos grandes como un monitor o soportes más pequeños como los de un móvil. **Nuestra visualización se puede adaptar** a diferentes resoluciones.

Compatible con estándares

Es compatible con estándares como JavaScript, HTML5 y *Cascading Style Sheets* (CSS3), por lo que funciona en diferentes exploradores y diferentes dispositivos.

Como complemento a la definición anterior, podríamos decir que, a alto nivel, D3 es una librería que se caracteriza por las siguientes **funcionalidades**:

- ▶ *Loading*: carga información en la memoria de tu explorador.
- ▶ *Binding*: añade información a elementos existentes en tu HTML, vinculando los datos y los objetos HTML.
- ▶ *Transforming*: es capaz de leer datos y dar una respuesta visual basada en los datos. Para ello puede transformar el código HTML.
- ▶ *Transitioning*: basándose en la entrada del usuario, puede reaccionar y dar una respuesta diferente.

Ideas clave

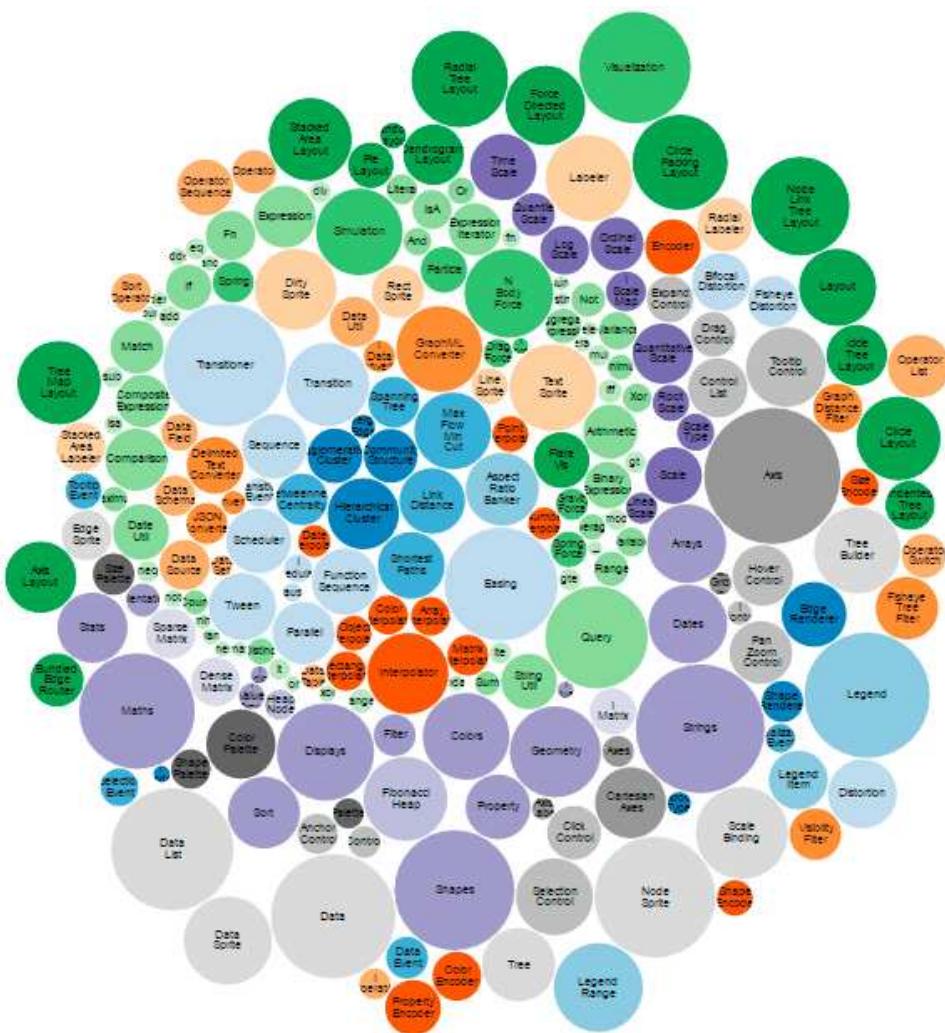


Figura 3. Ejemplo de visualización D3: Bubble Chart .Fuente:<https://www.tutorialsteacher.com/d3js/what-is-d3js>

Instalación

Para empezar a utilizar D3 no es necesario instalar ningún tipo de *software*. D3 es una librería *open source* por lo que está disponible de forma libre en su propia página web.

Accede a la página oficial de D3:

<https://d3js.org/>

Basta con llamar a la librería JavaScript de D3 desde nuestro propio código fuente.

Se puede hacer de dos formas distintas:

- ▶ Incluir la librería D3 desde nuestro ordenador: descargamos la última versión disponible en d3js.org a través del fichero d3.zip. Lo descomprimimos y extraemos la librería, ubicándola en el directorio local que queramos para poder llamarla desde nuestro código en JavaScript.
- ▶ Incluir la librería D3 desde CDN (*Content Delivery Network*): enlazamos a la última versión de la librería con el siguiente código:

En los ejemplos utilizaremos la versión que se ofrece en la web oficial:

<http://d3js.org/d3.v3.min.js>

Y este código será la base de nuestros ejercicios:

Código base para nuestros ejercicios

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

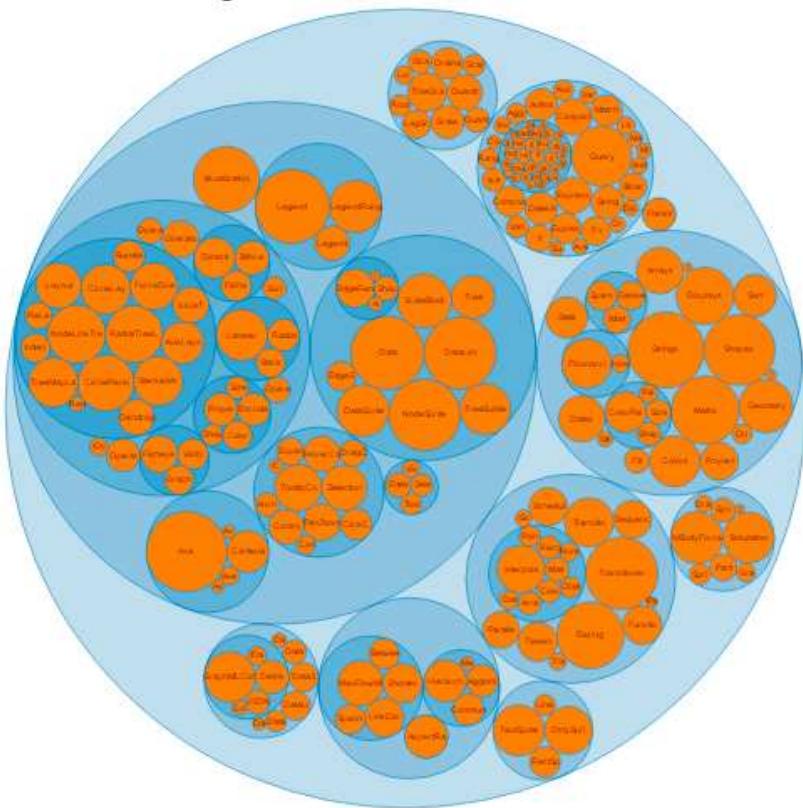


Figura 4. Ejemplo de visualización D3: *Circle Packing*. Fuente:<https://www.tutorialsteacher.com/d3js/what-is-d3js>

3.3. Elementos básicos de D3.js. Generando elementos HTML

Empezaremos con un ejemplo fácil. Hemos dicho que una de las características de D3 es que es capaz de transformar el código HTML y vincular datos con objetos HTML. ¿Cómo podemos crear un objeto HTML y añadir un texto con D3?

```
<html>
<head>
<script language="javascript" type="text/javascript"
src="http://d3js.org/d3.v3.min.js"></script>
<script type="text/javascript">
function draw(){
d3.select("body").append("p").text("Párrafo nuevo");
}
</script>
</head>
<body onload="draw()">
</body>
</html>
```

Si copiamos este texto en nuestro editor de código veremos que lo que hace es **generar un nuevo elemento** en nuestro HTML. Añade un párrafo nuevo cuando el HTML está cargado, , que llama a su vez a la función , que es la que realiza la acción en sí.

La **sintaxis de D3** sigue el concepto de encadenamiento o *chaining*. Se trata de concatenar varias funciones una seguida de otra, también es habitual utilizarla en Java, por ejemplo.

Pero ¿qué es lo que hace D3 con esta sentencia HTML?

- ▶ : selecciona el elemento body de nuestro HTML.
- ▶ : indica que añadirá algo a este elemento y define qué tipo de elemento HTML.

- ▶ : Define el texto que añadirá, es decir, el contenido texto.

De hecho, tendría el mismo efecto que utilizar lo siguiente:

```
var body = d3.select("body");
var p = body.append("p");
p.text("Párrafo nuevo");
```

3.4. Trabajando con datos reales y elementos en el HTML

Ahora veremos cómo trabajar con elementos dentro del HTML.

```
<html>
<head>
<script language="javascript" type="text/javascript"
src="http://d3js.org/d3.v3.min.js"></script>
<script type="text/javascript">
function draw(){
d3.select("p#target").text("Añadiendo contenido");
}
</script>
</head>
<body onload="draw()">
<p id="target"></p>
</body>
</html>
```

Esta estructura es muy parecida a la versión anterior, pero hemos utilizado las etiquetas o elementos habituales en HTML con las que hemos estado trabajado en temas anteriores.

Modificar elementos HTML y atributos

Pero D3 no solo nos permite añadir texto, sino también modificar los elementos HTML y sus atributos.

Añadir un atributo

Por ejemplo, queremos añadirle un atributo a nuestro elemento p llamado y asignarle el valor “ ”. Seguidamente lo imprimimos por pantalla. Si miramos nuestro contenido, el resultado final en el HTML es algo muy simple: nos muestra el texto “Contenido: primer_descriptor”. Y si miramos el código HTML en la consola de nuestro

explorador, podemos ver algo parecido a esto:

```
function draw(){
d3.select("p#target").text("Contenido: ");
// Establece atributo "descr" con el valor "primer descriptor" en
el elemento p
d3.select("p").attr("descr", "primer_descriptor");
// Obtiene el atributo "descr" en el elemento p
d3.select("p#target").text(d3.select("p").text()+d3.select("p").attr("descr"));

}
```

Si miramos nuestro contenido, el resultado final en el HTML es algo muy simple: nos muestra el texto . Y si miramos el código HTML en la consola de nuestro explorador, podemos ver algo parecido a esto:

```
<html>
<head>...</head>
<body onload="draw()">
<p id="target" descr="primer_descriptor">Contenido:
primer_descriptor</p>
</body>
</html>
```

Como puedes observar, hemos añadido un atributo a nuestro elemento en el propio código HTML.

Uso de estilos

Otro uso interesante de los elementos HTML a través de D3 es el uso de los estilos y lo podemos hacer a través de la función .

Por ejemplo, cambiarle el tamaño de la fuente:

```
d3.select("p#target").style("font-size", function(){return
"40px";});
```

Ahora mismo, dentro del código de la función solo devuelve 40px, pero podríamos

asignarle un valor dinámico dependiendo de alguna otra variable o del valor de un dato, lo que demuestra las enormes posibilidades que ofrece D3 al poder crear dinámicamente objetos HTML.

Modificar varios elementos HTML al mismo tiempo

Ahora aplicaremos estos conceptos en varios elementos HTML al mismo tiempo.

Para ello, utilizaremos uno de tantos ejemplos que podemos encontrar en el libro *Data Visualization with D3.js Cookbook* (Zhu, 2013) en el que los ejemplos son de libre acceso y se pueden descargar. El ejemplo es el siguiente:

```
<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 draw(){
d3.selectAll("div")
.attr("class", "red box");
}
</script>
</head>
<body onload="draw()">
<div></div>
<div></div>
<div></div>
</body>
</html>
```

La hoja de estilo a utilizar está definida en el archivo styles.css. En él, la clase `.box` y la clase `red` están definidas de la siguiente forma:

```
.box {
width: 200px;
height: 200px;
margin: 40px;
```

```
float: left;
text-align: center;
border: #969696 solid thin;
padding: 5px;
}
.red {
background-color: #e9967a;
color: #f0f8ff;
}
.blue {
background-color: #add8e6;
color: #f0f8ff;
}
```

Aplicando las clases `y` a los diferentes elementos se obtiene el siguiente resultado:

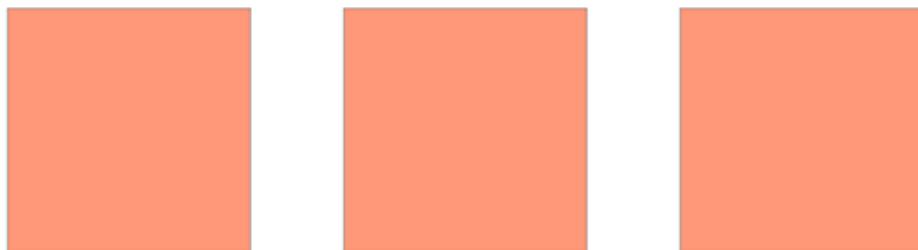


Figura 5. Resultado de dibujar 3 cajas.

Si queremos **iterar** sobre cada elemento para hacer algo diferente, lo podemos hacer de la siguiente manera:

```
d3.selectAll("div")
.attr("class", "red box")
.each(function (d, i) {
d3.select(this).append("h1").text(i);
});
```

Aquí resaltaremos lo que se diferencia de lo anterior:

```
each(function (d, i) {
```

```
d3.select(this).append("h1").text(i);
```

La función `itera` sobre los diferentes elementos, siendo `d` el elemento en sí, es decir, el elemento `div` referenciado dentro de la función como `this`. Y lo que hacemos es añadir un elemento `h1` con el número de la iteración.

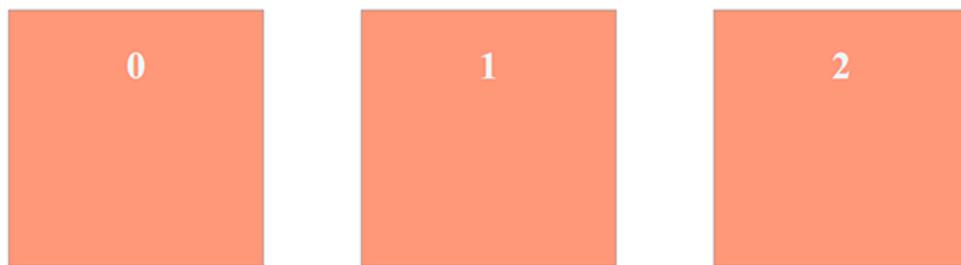


Figura 6. Resultado de dibujar 3 cajas con texto.

Subelementos

Podemos trabajar también con **subelementos**. En el siguiente ejemplo, veremos cómo seleccionar un subelemento dentro de otro elemento. Cambiemos nuestro código por lo siguiente:

```
<section id="section1">
<div>
<p>blue box</p>
</div>
</section>
<section id="section2">
<div>
<p>red box</p>
</div>
</section>
```

Y el código de la función por este:

```
d3.select("#section1 > div")
.attr("class", "caja azul");
```

```
d3.select("#section2")
  .select("div")
  .attr("class", "caja roja");
```

En el código anterior podemos observar dos maneras diferentes de seleccionar un subelemento:

- ▶ identificador > elemento html:

```
d3.select("#section1 > div")
```

- ▶ doble select:

```
d3.select("#section2")
```

```
  .select("div")
```

El código completo queda de la siguiente forma:

```
<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 draw(){
d3.select("#section1 > div")
  .attr("class", "blue box");
d3.select("#section2")
  .select("div")
  .attr("class", "red box");
}
</script>
</head>
<body onload="draw()">
<section id="section1">
<div>
<p>caja azul</p>
</div>
</section>
```

```
<section id="section2">
<div>
<p>caja roja</p>
</div>
</section>
</body>
</html>
```

Y el resultado final es el siguiente:

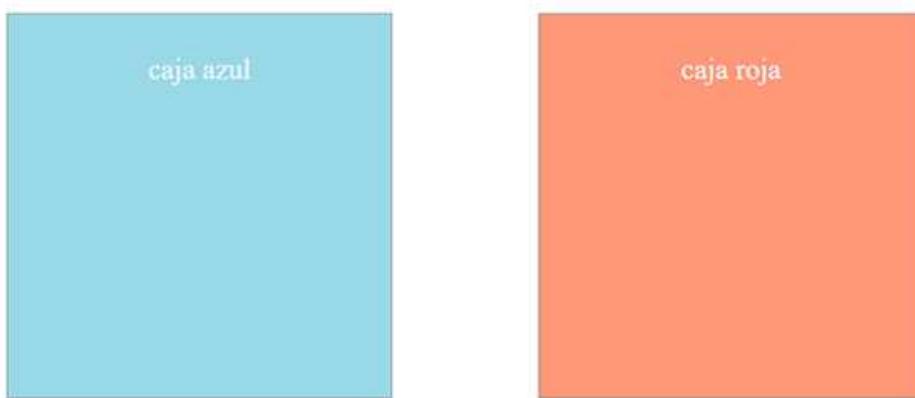


Figura 7. Resultado dibujar 2 cajas de diferente color y con texto.

Dando un paso más, también podríamos reemplazar toda la sección y dejarla completamente limpia . Para ello incluimos el código D3 siguiente dentro de la función :

```
<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 draw(){
var body = d3.select("body");
body.append("section")
.attr("id", "section1")
.append("div")
.attr("class", "blue box")
```

```
.append("p")
.text("caja azul dinámica");
body.append("section")
.attr("id", "section2")
.append("div")
.attr("class", "red box")
.append("p")
.text("caja roja dinámica");
d3.select("#section1 > div")
.attr("class", "blue box");
d3.select("#section2")
.select("div")
.attr("class", "red box");
}
</script>
</head>
<body onload="draw()">
</body>
</html>
```

¿Cuáles son las principales diferencias?

Primero creamos una variable y le asignamos el resultado de la función . Esto es lo mismo que hemos hecho al principio del tema.

Para crear la primera caja añadimos un elemento . Le añadimos un atributo identificador llamado . A este elemento , le añadimos un elemento y le asignamos la clase . A este elemento le añadimos un elemento párrafo o y le asignamos el texto caja azul dinámica. La caja roja sigue el mismo patrón de creación.

3.5. Ventajas de D3.js

Para terminar este tema, enunciaremos algunas de las ventajas del uso de D3 como herramienta de visualización.

Hemos visto a través de los ejemplos que es necesario conocimientos de JavaScript, aunque el esfuerzo necesario tiene su recompensa:

- ▶ D3.js es una biblioteca de JavaScript. Por lo tanto, puede ser usada con cualquier marco JS de su elección como Angular.js, React.js, etc...
- ▶ D3 se centra en los datos, por lo que es la herramienta más apropiada y especializada para la visualización de datos.
- ▶ D3 es de código abierto, te permite trabajar con el código fuente y añadir tus propias características.
- ▶ Funciona con estándares web, por lo que no se necesita ninguna otra tecnología o *plugin* que no sea un navegador para hacer uso de D3.
- ▶ D3 trabaja con estándares web como HTML, CSS y SVG, no se requiere ninguna nueva herramienta de aprendizaje o depuración para trabajar en D3.
- ▶ D3 proporciona un control completo sobre su visualización para personalizarla de la manera que se deseé. Esto le da una ventaja sobre otras herramientas populares como Tableau o QlikView para usos específicos.
- ▶ Dado que D3 es ligero y funciona directamente con los estándares web, es extremadamente rápido y funciona bien con grandes conjuntos de datos.

En definitiva, una potente librería que, aunque requiere una curva de aprendizaje en términos de programación general, a cambio aporta una gran flexibilidad y posibilidades casi infinitas.

3.6. Referencias bibliográficas

100 frases. *Frase de Barbara Sher*. <https://100frases.com/autor/barbara-sher/puedes-aprender-cosas-nuevas-en-cualquier-momento--5b6e09f373e77aa867125151>

TutorialsTeacher. *D3.js Tutorial*. <https://www.tutorialsteacher.com/d3js>

Zhu, N. Q. (2013). *Data Visualization with D3.js Cookbook*. [S. l.]: Packt Publishing. <https://github.com/NickQiZhu/d3-cookbook>

D3 for the Impatient: Interactive Graphics for Programmers and Scientists

Janert, P. K. (2013). *D3 for the Impatient: Interactive Graphics for Programmers and Scientists*. [S. l.]: O'Reilly.

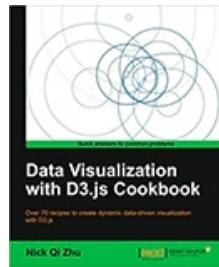


Este libro está recomendado para quienes entienden lo básico de HTML, CSS y JavaScript y quieran explorar rápidamente la extensa, pero a menudo abrumadora, documentación de referencia sobre D3.js.

Philipp K. Janert, autor de *Data Analysis with Open Source Tools*, proporciona una concisa hoja de ruta para esta librería, incluyendo sus convenciones y conceptos fundamentales. *D3.js para los impacientes* es conciso y completo.

Data Visualization with D3.js Cookbook

Zhu, N. Q. (2013). *Data Visualization with D3.js Cookbook*. [S. I.]: Packt Publishing. <https://github.com/NickQiZhu/d3-cookbook>



Convierte tus datos digitales en gráficos dinámicos con este libro repleto de guías y ejemplos útiles sobre la biblioteca de JavaScript D3, que te ayudarán a resolver problemas reales de visualización.

Data Driven Documents D3.js: Tips and Tricks

Maclean, M. (S. f.). *Data Driven Documents D3.js: Tips and Tricks* (v.3.x) [S. l.]: Leanpub. <https://leanpub.com/D3-Tips-and-Tricks/read#leanpub-auto-what-do-you-need-to-get-started>



Este libro, escrito por Malcolm Maclean y de libre acceso, ofrece consejos y trucos para utilizar d3.js. Es un libro que te ayudará a empezar, pero también incluye algunas tareas avanzadas.

Creando un mapa sencillo con D3

Morales, A. (15 de marzo de 2016). Creando un mapa sencillo con D3. *Mapping GIS*. <https://mappinggis.com/2015/03/creando-un-mapa-sencillo-con-d3/>

Este artículo explica cómo crear un mapa sencillo en D3 que muestre los países del mundo.



Learn D3.js in 5 minutes

Nehal, S. (6 de abril de 2018). Learn D3.js in 5 minutes. *Free Code Camp*. <https://www.freecodecamp.org/news/learn-d3-js-in-5-minutes-c5ec29fb0725/>

Introducción a la creación de representaciones visuales con tus datos.

6 APRIL 2018 / #JAVASCRIPT

Learn D3.js in 5 minutes

- 1.** ¿Qué significa el concepto *chaining* en D3?
 - A. Las funciones se pueden concatenar una detrás de otra separándolas con un punto.
 - B. Las funciones dependen unas de otras, y solo puedes llamarlas de una manera seguida.
 - C. Las funciones permiten crear elementos HTML.
 - D. Ninguna de las anteriores.

- 2.** Si ejecutamos la sentencia :
 - A. Seleccionamos el elemento p con identificador target en el HTML y añadimos el texto indicado entre comillas.
 - B. Seleccionamos el elemento p con identificador target en el HTML y reemplazamos el texto existente con el indicado entre comillas.
 - C. Seleccionamos el elemento target con identificador p.
 - D. Ninguna de las anteriores.

- 3.** Si ejecutamos la sentencia
 - A. Seleccionamos el elemento del HTML, añadimos un elemento p y le asignamos el texto entre comillas.
 - B. Seleccionamos el elemento del HTML, con identificador y le asignamos el texto entre comillas.
 - C. No tiene una sintaxis correcta y no funciona.
 - D. Ninguna de las anteriores.

4. Con D3, ¿podemos modificar las clases CSS de los elementos HTML?

 - A. Sí.
 - B. No, porque figuran un fichero independiente.
 - C. No, porque D3 solo me permite trabajar con los datos.
 - D. No, porque solo puedo modificar el código HTML
5. Con D3, ¿podemos modificar el estilo de un elemento HTML como, por ejemplo, el tamaño de la fuente, con una sentencia tipo ?

 - A. Sí.
 - B. No, porque las características de los objetos residen en las hojas de estilo.
 - C. No, porque no existe una forma de modificar solo un identificador de estilo.
 - D. No, porque los elementos de las hojas de estilo no son accesibles por D3.
6. ¿Qué hace la siguiente sentencia: ?

 - A. Devuelve todos los elementos div del HTML con el atributo clase que contenga .
 - B. Asigna a todos los elementos div del HTML la clase .
 - C. Asigna a un elemento div del HTML la clase .
 - D. Ninguna de las anteriores.
7. D3 permite trabajar tanto con los datos a visualizar como con elementos HTML:

 - A. No, solo con datos.
 - B. No, solo con elementos HTML.
 - C. No, porque los elementos u objetos HTML deben codificarse antes de su visualización por el navegador.
 - D. Sí. Es una de sus principales ventajas como herramienta de visualización.

8. ¿Qué hace la siguiente sentencia: ?

- A. Itera por todos los elementos y les añade un elemento con el texto , donde es el contador de la iteración.
- B. Itera por todos los elementos y les añade un elemento con el texto , donde es el texto que el elemento HTML contiene.
- C. Itera por todos los elementos y les añade un elemento con el texto , donde es una constante fija previamente definida.
- D. Ninguna de las anteriores.

9. Identifica si obtenemos el mismo resultado con la sentencia que con el siguiente código:

```
var body = d3.select("body");
var p = body.append("p");
p.text("Párrafo nuevo");
```

- A. Sí.
- B. No, porque la primera sentencia no es correcta.
- C. No, porque aunque la primera sentencia es correcta, el código siguiente no lo es.
- D. No, porque tanto la línea de código individual como el grupo de código son incorrectos.

10. ¿Qué hace la sentencia d ?

- A. Le asigna vacío al elemento con identificador .
- B. Devuelve el contenido del elemento con identificador .
- C. Devuelve el contenido del elemento , siendo este de formato texto.
- D. Le asigna un texto al elemento .

Herramientas de Visualización

Tema 4. D3.js. Datos, SVG y gráficas

Índice

[Esquema](#)

[Ideas clave](#)

[4.1. Introducción y objetivos](#)

[4.2. Trabajar con diferentes estructuras de datos JSON y CSV](#)

[4.3. Generar y dibujar con SVG](#)

[4.4. Bar Chart y Scatter Plot desde cero](#)

[4.5. Referencias bibliográficas](#)

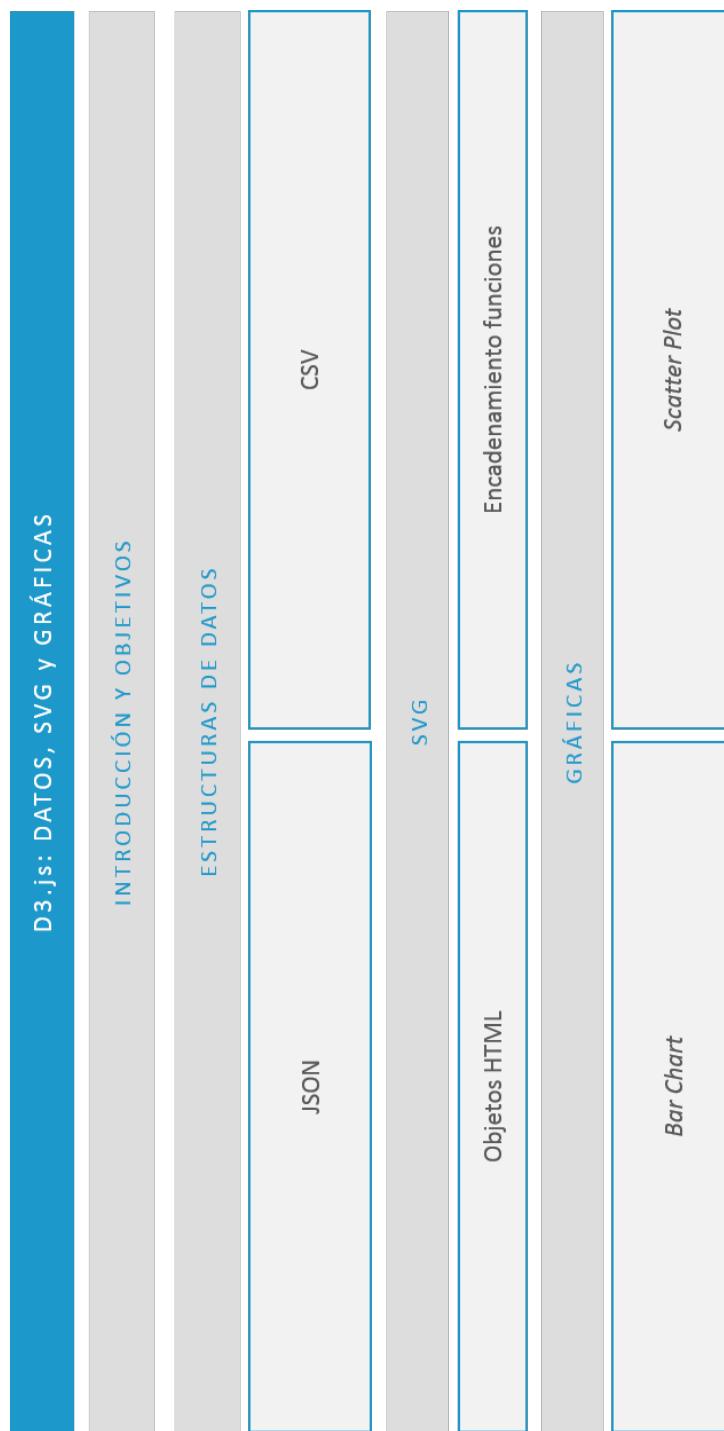
[A fondo](#)

[Interactive Data Visualization for the Web](#)

[Teaching an old dog new tricks. How data visualization & design can be used by everyone](#)

[Film Brazil](#)

[Test](#)



4.1. Introducción y objetivos

Ya hemos comenzado a experimentar con D3 en el tema anterior. Hemos visto la forma de trabajar con esta biblioteca, pero todavía no hemos creado ninguna gráfica.

En este tema introduciremos los elementos más importantes para empezar a dibujar y para ello, trabajaremos con ficheros CSV o JSON. También abordaremos los elementos SVG (*Scalable Vector Graphics*) en HTML y empezaremos a dibujar de una manera sencilla un *Bar Chart* y un *Scatter Plot*, todo con D3 y con elementos HTML.

Jugaremos con los estilos de las gráficas y anotaremos ciertos textos en los gráficos, pero algo simple, ya tendremos tiempo para hacer cosas más complejas.

Comenzamos este tema con una frase de Confucio:

«Life is really simple, but men insist on making it complicated» (Brainy Quote).

Siguiendo la línea del tema anterior, aconsejamos asentar primero los conceptos. Si saltamos rápidamente a intentar modificar visualizaciones, el aprendizaje puede complicarse, no hay nada malo en empezar lento pero seguro, aunque se tengan conocimientos de programación o de JavaScript, con D3 es fácil perderse.

Estudiar este tema no es diferente a los anteriores, recomendamos practicar con todas las instrucciones del código. Elige Brackets u otra herramienta para que puedas repetir los pasos. Además, el apartado sobre las estructuras de datos JSON y CSV es importante, así que insistimos en la relevancia de asimilar su contenido. Entender cómo funciona la gestión de datos permitirá modificar futuras visualizaciones de manera más ágil.

Por otro lado, en el resto de apartados del tema se asientan conceptos que luego podremos ir desarrollando con más facilidad, por lo que se aconseja probar todo el

código.

4.2. Trabajar con diferentes estructuras de datos JSON y CSV

CSV

Empezaremos creando un archivo CSV en Brackets. Lo llamaremos «comida.csv»:

- ▶ Comida,Preferencia.
- ▶ Manzana,9.
- ▶ Huevos fritos,10.
- ▶ Ensalada,5.
- ▶ Galleta,2.
- ▶ Leche,8.

¿Cómo cargamos el archivo con D3?:

```
<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(){
d3.csv("comida.csv", function(data) {
data.forEach(function(d){
d3.select("body").append("p").text(d.Comida + " " + d.
Preferencia);
});
});
}
</script>
</head>
<body onload="read()">
```

```
</body>
</html>
```

Analicemos lo importante:

```
d3.csv("comida.csv", function(data) {
  data.forEach(function(d){
    d3.select("body").append("p").text(d.Comida + " " + d.
      Preferencia);
  })
})
```

La función CSV acepta **dos atributos**:

- ▶ El primero es la **ruta al archivo**, es decir, que si el archivo no está en la misma carpeta del HTML, posiblemente dará error. Es importante recordar estos pequeños detalles.
- ▶ La segunda es la **función**, que ya hemos ido viendo en otros temas. Si la función no encuentra el archivo CSV, la función nunca será llamada.

Lo que hacemos a continuación es imprimir todos los valores de nuestro CSV, observa que no es una matriz de valores. Accede a los valores iterando por las filas y luego llama a los valores basándote en las cabeceras.

JSON

De manera muy similar, podemos trabajar con archivos JSON. Crearemos un ejemplo de coordenadas. El nuevo archivo lo llamaremos «coordenadas.json».

```
[{
  "x_axis": 30,
  "y_axis": 30,
  "radius": 20,
  "color": "green"
}, {
  "x_axis": 70,
  "y_axis": 70,
```

```
"radius": 20,  
"color": "purple"  
, {  
"x_axis": 110,  
"y_axis": 100,  
"radius": 20,  
"color": "red"  
}]
```

Leeremos el objeto y lo recorreremos de forma muy similar al CSV:

```
d3.json("coordenadas.json", function(data) {  
data.forEach(function(d){  
d3.select("body").append("p").text(d.x_axis + " " + d.radius+ " "  
+ d.y_axis + " " + d.color);  
})  
})
```

En este nuevo ejemplo, fíjate que utilizamos la función `d3.json` para leer el fichero en lugar de la función `csv.json`.

4.3. Generar y dibujar con SVG

D3 es mucho más útil cuando lo utilizamos para gestionar objetos gráficos como los elementos SVGs (*Scalable Vector Graphic*). También podemos crear visualizaciones con elementos div, pero es mucho más efectivo trabajar con estos elementos para evitar problemas entre diferentes tipos de dispositivos y exploradores web y asegurar así, la **máxima compatibilidad**.

Empezamos con un ejemplo. Pon en el `<div>` del HTML el siguiente código:

```
<svg width="50" height="50">
<circle cx="25" cy="25" r="22" fill="blue" stroke="gray" stroke-width="2"/>
</svg>
```

Obtendremos el siguiente resultado:



Figura 1. Resultado objeto SVG.

Se recomienda siempre generar los elementos SVG con sus **atributos** y .

La unidad por defecto es píxel, por esa misma razón veremos que las medidas de nuestro ejemplo son 50 y no 50px. También podemos jugar con la posición dentro de nuestra pantalla. Recuerda que el **elemento 0,0** siempre es la esquina superior izquierda.

```
<rect x="100" y="0" width="200" height="50"/>
```

Si te fijas en la diferencia, cuando hemos dibujado un **círculo**, las coordenadas `cx` y `cy` indican las coordenadas del centro del círculo, además se define un radio para

definir el tamaño, mientras que en el **rectángulo**, indicarían la posición de la esquina superior izquierda.

Podemos dibujar una **elipse**:

```
<ellipse cx="250" cy="25" rx="100" ry="25"/>
```

También es importante saber cómo dibujar una **línea**. Aquí definiremos el punto inicial y el punto final y elemento que les une: .

```
<line x1="0" y1="0" x2="500" y2="50" stroke="black"/>
```

Podemos **renderizar** elementos gráficos:

```
<text x="250" y="25" font-family="serif" font-size="25"  
fill="gray"></text>
```

Como en casi cualquier elemento HTML, puedes definir el **estilo**:

- ▶ : si lo quieres rellenar. Puedes utilizar los mismos valores que en CSS para los colores.
- ▶ : el color de la línea.
- ▶ : anchura de la línea.
- ▶ : un número entre 0.0 y 1.0.

También se puede **utilizar CSS**. Por ejemplo, especificamos la clase en el SVG:

```
<circle cx="25" cy="25" r="22" class="pumpkin"/>
```

Y la definimos:

```
.pumpkin {  
fill: yellow;  
stroke: orange;
```

```
stroke-width: 5;  
}
```

Es importante saber que en SVG no existe el concepto capas ni manera de definir la posición del eje z, ya que nos movemos en un plano bidimensional. Por ejemplo, si queremos causar un efecto como el de la figura 2:

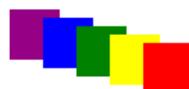


Figura 2. Rectángulos superpuestos en plano bidimensional.

Deberemos jugar con el orden y la posición de la x y la y:

```
<rect x="0" y="0" width="30" height="30" fill="purple"/>  
<rect x="20" y="5" width="30" height="30" fill="blue"/>  
<rect x="40" y="10" width="30" height="30" fill="green"/>  
<rect x="60" y="15" width="30" height="30" fill="yellow"/>  
<rect x="80" y="20" width="30" height="30" fill="red"/>
```

Si quieres jugar con las **transparencias**, puedes utilizar los colores en formato RGBA, donde el último valor indica la transparencia:

```
<circle cx="25" cy="25" r="20" fill="rgba(128, 0, 128, 1.0)"/>  
<circle cx="50" cy="25" r="20" fill="rgba(0, 0, 255, 0.75)"/>  
<circle cx="75" cy="25" r="20" fill="rgba(0, 255, 0, 0.5)"/>  
<circle cx="100" cy="25" r="20" fill="rgba(255, 255, 0, 0.25)"/>  
<circle cx="125" cy="25" r="20" fill="rgba(255, 0, 0, 0.1)"/>
```

Da el siguiente resultado:



Figura 3. Círculos superpuestos con transparencia.

Pero también podemos jugar todavía con el elemento de la **opacidad** que hemos

comentado antes. Por ejemplo:

```
<circle cx="25" cy="25" r="20" fill="purple" stroke="green"
stroke-width="10"
opacity="0.9"/>
<circle cx="65" cy="25" r="20" fill="green" stroke="blue" stroke-
width="10"
opacity="0.5"/>
<circle cx="105" cy="25" r="20" fill="yellow" stroke="red" stroke-
width="10"
opacity="0.1"/>
```

Da como resultado:



Figura 4. Círculos superpuestos con transparencia y opacidad.

4.4. Bar Chart y Scatter Plot desde cero

Una vez explicados los elementos SVG, explicaremos cómo hacer un *Bar Chart* y un *Scatter Plot* desde cero. En un principio no utilizaremos elementos SVG, sino que usaremos elementos div para trabajar con ellos. Trabajaremos con una estructura simple de datos:

```
var dataset = [ 5, 10, 15, 20, 25 ];
```

Como ejercicio intenta trabajar con la función de lectura del CSV y sigue los pasos con el CSV en lugar de este array.

Bar Chart

¿Cómo podemos dibujar una barra con el uso exclusivo del elemento ?

Prueba este código:

```
<div style="display: inline-block;  
width: 20px;  
height: 75px;  
background-color: teal;">  
</div>
```

Como podemos observar, lo logramos definiendo el estilo como y se obtiene lo siguiente:



Figura 5. Barra creada con elementos div.

Para poder aplicar este estilo a diferentes , crearemos nuestra clase en CSS:

```
div.bar {  
display: inline-block;  
width: 20px;  
height: 75px; /* We'll override height later */  
background-color: teal;  
}
```

Por lo tanto, ahora solo tendremos que referenciarlo de la siguiente manera obteniendo el mismo resultado:

```
<div class="bar"></div>
```

Intentamos hacer lo mismo con D3, por cada elemento del *array dataset* que hemos creado antes:

```
function read(){  
d3.select("body").selectAll("div")  
.data(dataset)  
.enter()  
.append("div")  
.attr("class", "bar");  
}
```

Verás que ahora tenemos cinco barras con la misma altura y concatenadas unas a otras, lo que no es muy útil, ya que la altura debería variar según el contenido del *array*.

Para conseguir el comportamiento deseado hemos de sobrescribir el valor de nuestro CSS. Para ello solo tenemos que añadir una pequeña sentencia después del código D3:

```
function read(){  
d3.select("body").selectAll("div")  
.data(dataset)
```

```
.enter()
.append("div")
.attr("class", "bar")
.style("height", function(d) {
return d + "px";
})
}
```

Si quieres ver el comportamiento con **más valores** en el *array*, puedes probar a generar el *array* con valores aleatorios.

```
var dataset = []; //Inicializar array vacío
for (var i = 0; i < 10; i++) { //Loop 10 veces
var newNumber = Math.floor(Math.random() * 30); //Números
aleatorios (0-30)
dataset.push(newNumber); //Añadir al array
}
```

Este es el ejemplo que se da para 10 valores. Solo tienes que modificar ese número si quieres más o menos valores.

El código completo quedaría de la siguiente manera:

```
<html>
<head>
<script language="javascript" type="text/javascript"
src="http://d3js.org/d3.v3.min.js"></script>
<style>
.bar {
display: inline-block;
width: 50px;
height: 75px; /* Lo sobreescribiremos */
background-color: teal;
}
</style>
<script>
var dataset = []; //Inicializar array vacío
for (var i = 0; i < 10; i++) { //Loop 10 veces
var newNumber = Math.floor(Math.random() * 30);
```

```
dataset.push(newNumber); //Añadir al array
}
function read(){
d3.select("body").selectAll("div")
.data(dataset)
.enter()
.append("div")
.attr("class", "bar")
.style("height", function(d) {
return d + "px";
})
}
</script>
</head>
<body onload="read()">
</body>
</html>
```

El resultado con 10 valores aleatorios es el siguiente:



Figura 6. Barras aleatorias en D3.

Ya lo sabemos hacer con elementos . ¿Qué tal si lo intentamos con elementos SVG?

¿Cómo podemos dibujar una barra con elementos SVG?

Primero definiremos el **tamaño** de nuestro SVG:

```
//Width and height
var w = 500;
var h = 100;
```

Luego crearemos nuestro **elemento** SVG:

```
//Create SVG element
var svg = d3.select("body")
```

```
.append("svg")
.attr("width", w)
.attr("height", h);
```

Podemos ver que lo que realmente crea el elemento SVG es la llamada a la función .
Luego se generan las barras creando rectángulos.

```
svg.selectAll("rect")
.data(dataset)
.enter()
.append("rect")
.attr("x", 0)
.attr("y", 0)
.attr("width", 20)
.attr("height", 100);
```

Si ejecutamos esto, veremos que solo tenemos una barra negra del mismo tamaño.

Hemos creado cinco barras, pero todas están en la misma X e Y, y todas tienen la misma altura. ¿Cómo lo arreglamos?

```
svg.selectAll("rect")
.data(dataset)
.enter()
.append("rect")
.attr("x", function(d, i) {
return i * (w / dataset.length);
})
.attr("y", 0)
.attr("width", w / dataset.length - 1)
.attr("height", function(d) {
return d;
});
```

Atención a lo que hacemos con los valores , y . El -1 es para hacer la separación entre barra y barra, pero si ejecutas esto en Brackets, ¿cuál será el problema?

El gráfico te sale invertido. Bueno hemos de recordar que empezamos a dibujar en la

coordenada 0,0 que está en la esquina superior izquierda, por lo que la altura definida al rectángulo es hacia abajo.



Figura 7. Barras aleatorias en D3 con SVG.

Para que todo salga bien, tienes que cambiar el valor de la "y" en cada elemento.

```
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; // Altura menos el valor del dato
  })
  .attr("width", w / dataset.length - 1)
  .attr("height", function(d) {
    return d;
});
```

El cambio que hacemos en la `y` no es para decirle que «dibuje» hacia arriba, sino que desplazamos la `y` para que, cuando empiece a dibujar, se queden todos alineados en el eje virtual de las `y`. Pero el dibujo sigue estando hacia abajo.

Añadiendo el atributo `fill`, le podemos dar un poco de **color**:

```
.attr("fill", "teal");
```

Hasta le podemos poner **gradiente**:

```
.attr("fill", function(d) {
  return "rgb(0, 0, " + (d * 10) + ")";
})
```

Y ahora pondremos el **valor encima de las barras**, añadiendo el valor que le toca:

```
svg.selectAll("text")
  .data(dataset)
  .enter()
  .append("text")
  .text(function(d) {
    return d;
  })
  .attr("x", function(d, i) {
    return i * (w / dataset.length);
  })
  .attr("y", function(d) {
    return h - (d);
 });
```

El resultado final sería algo tal que así:

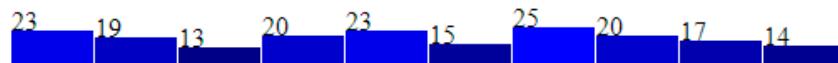


Figura 8. Barras aleatorias en D3 con SVG, gradiente de color y valor.

Scatter Plot

¿Cómo podríamos hacer un Scatter Plot de manera similar? Trabajaremos con otro tipo de *array*:

```
var dataset = [
  [5, 20], [480, 90], [250, 50], [100, 33], [330, 95], [410, 12],
  [475, 44],
  [25, 67], [85, 21], [220, 88]
];
```

La función es muy parecida a la que utilizamos para el *Bar Chart*:

```
svg.selectAll("circle") // <-- Cambiamos donde antes ponía "rect"
  .data(dataset)
```

```
.enter()
.append("circle")
.attr("cx", function(d) {
  return d[0];
})
.attr("cy", function(d) {
  return d[1];
})
.attr("r", 5);
```

Y ya deberíamos tener un resultado similar a este:



Figura 9. Scatter Plot.

Evidentemente, nos queda aún dibujar los ejes y demás elementos, pero todo es comenzar.

Para ejercitarse con este tema, se recomienda hacer este proceso con un archivo CSV y otro JSON para ambos casos. Si tienes problemas con los navegadores en local, que no permiten acceder a ficheros externos, introduce los datos en el código, aunque no sea una situación tan real como cuando tenemos a nuestra disposición un servidor donde publicar nuestros gráficos.

Por último, recomendamos los siguientes tutoriales prácticos:

- ▶ w3schools.com | SVG Tutorial: https://www.w3schools.com/graphics/svg_intro.asp
- ▶ W3C | SCALABLE VECTOR GRAPHICS (SVG): <https://www.w3.org/Graphics/SVG/>
- ▶ DesarrolloWeb.com | Qué es SVG: <https://desarrolloweb.com/articulos/que-es-svg.html>

4.5. Referencias bibliográficas

Brainy

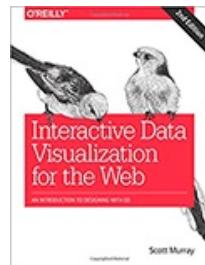
Quote. *Confucius*

Quotes.

https://www.brainyquote.com/quotes/confucius_104563

Interactive Data Visualization for the Web

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



Para este tema son interesante las secciones del libro que contienen información sobre los SVGs y explican cómo dibujar un *Bar Chart* y un *Scatter Plot*.

Teaching an old dog new tricks. How data visualization & design can be used by everyone

Morris, C. y Wocknitz, S. (2013). Teaching an old dog new tricks. How data visualization & design can be used by everyone. *Proceedings of Annual Conference 2013.* http://www.samra.co.za/wp-content/uploads/2013/05/Wocknitz-Morris_Teaching-an-old-dog-new-tricks-Research-Paper.pdf

En este documento se muestra cómo la información puede ser mucho más accesible a través de la visualización y el diseño.

Film Brazil

Pommella, D. y Carelli, R. (2013). *Film Brazil* [Vídeo]: Vimeo. <https://vimeo.com/68577610>

Un vídeo, ejemplo de inspiración, sobre cómo utilizar infografía.



1. ¿Podemos utilizar clases CSS en SVG?

 - A. Sí.
 - B. No, no son compatibles con HTML.
 - C. No, no son necesarias.
 - D. No, porque D3 no incorpora clases CSS.

2. Si queremos dibujar una elipse, ¿debemos utilizar el SVG elemento de e indicar que el radio es variante?

 - A. Sí.
 - B. No, requiere el *elemento* y cuatro parámetros: cx, cy, rx, ry.
 - C. No, requiere el *elemento* y dos parámetros: cx, cy.
 - D. No, requiere el *elemento* y tres parámetros: cx, cy, r.

3. Para dibujar una línea debemos definir los puntos de partida y de fin, y qué tipo de línea queremos que les una:

 - A. Sí.
 - B. No.
 - C. No porque depende de si queremos líneas continuas o a trazos.
 - D. No existen líneas en D3.

4. Si ponemos el valor de la altura en negativo cuando vamos a dibujar un rectángulo, ¿el rectángulo se dibujará hacia arriba?

 - A. Sí, porque valores positivos apuntan hacia abajo y negativos hacia arriba.
 - B. No.
 - C. Sí, porque las coordenadas empiezan en 0,0.
 - D. Sí, porque las coordenadas tienen su origen en el margen superior izquierdo.

5. Cuando dibujamos con SVG, ¿qué diferencia hay entre cx y x?
 - A. cx indica el centro del círculo y x indica el comienzo a dibujar, por ejemplo, en un rectángulo la esquina superior izquierda.
 - B. cx indica el comienzo a dibujar, por ejemplo, en un rectángulo la esquina superior izquierda, y x indica el centro del círculo.
 - C. Ninguna.
 - D. No existen el parámetro x.
6. ¿Qué función hemos de utilizar para cargar un archivo CSV?
 - A.
 - B.
 - C.
 - D.
7. ¿Qué función hemos de utilizar para cargar un archivo JSON?
 - A.
 - B.
 - C.
 - D.

8. ¿Qué es lo que hace este código?

```
var dataset = [];
for (var i = 0; i < 50; i++) {
var newNumber = Math.random() * 30;
dataset.push(newNumber);
}
```

- A. Nos genera un *array* de 50 números del 0 al 30.
- B. Nos genera un *array* de 30 números del 0 al 50.
- C. Dibuja 50 números al azar.
- D. Dibuja 30 números del 0 al 50.

9. ¿Qué nos dibujará el siguiente código?

```
var dataset = [ 5, 10, 15, 20, 25 ];
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;
})
.attr("width", w / dataset.length - 1)
.attr("height", function(d) {
return d;
});
```

- A. Cinco barras con color gradiente azul con el texto encima de las barras.
- B. Diez barras con color gradiente azul con el texto encima de las barras.
- C. Cinco barras con la dirección invertida.
- D. Ninguna de las anteriores.

10. ¿Qué nos dibujará el siguiente código?

```
var dataset = [  
[5, 20], [480, 90], [250, 50], [100, 33], [330, 95],  
[410, 12], [475, 44], [25, 67], [85, 21], [220, 88]  
]  
  
svg.selectAll("circle")  
.data(dataset)  
.enter()  
.append("circle")  
.attr("cx", function(d) {  
return d[0];  
})  
.attr("cy", function(d) {  
return d[1];  
})  
.attr("r", 5);
```

- A. Diez círculos con radio 5 en posición *random* en nuestro SVG.
- B. Nueve círculos con radio 5 en posición definida por el dataset.
- C. Una gráfica *Scatter Plot* con diez barras definidas por el dataset.
- D. Ninguna de las anteriores.

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](#)

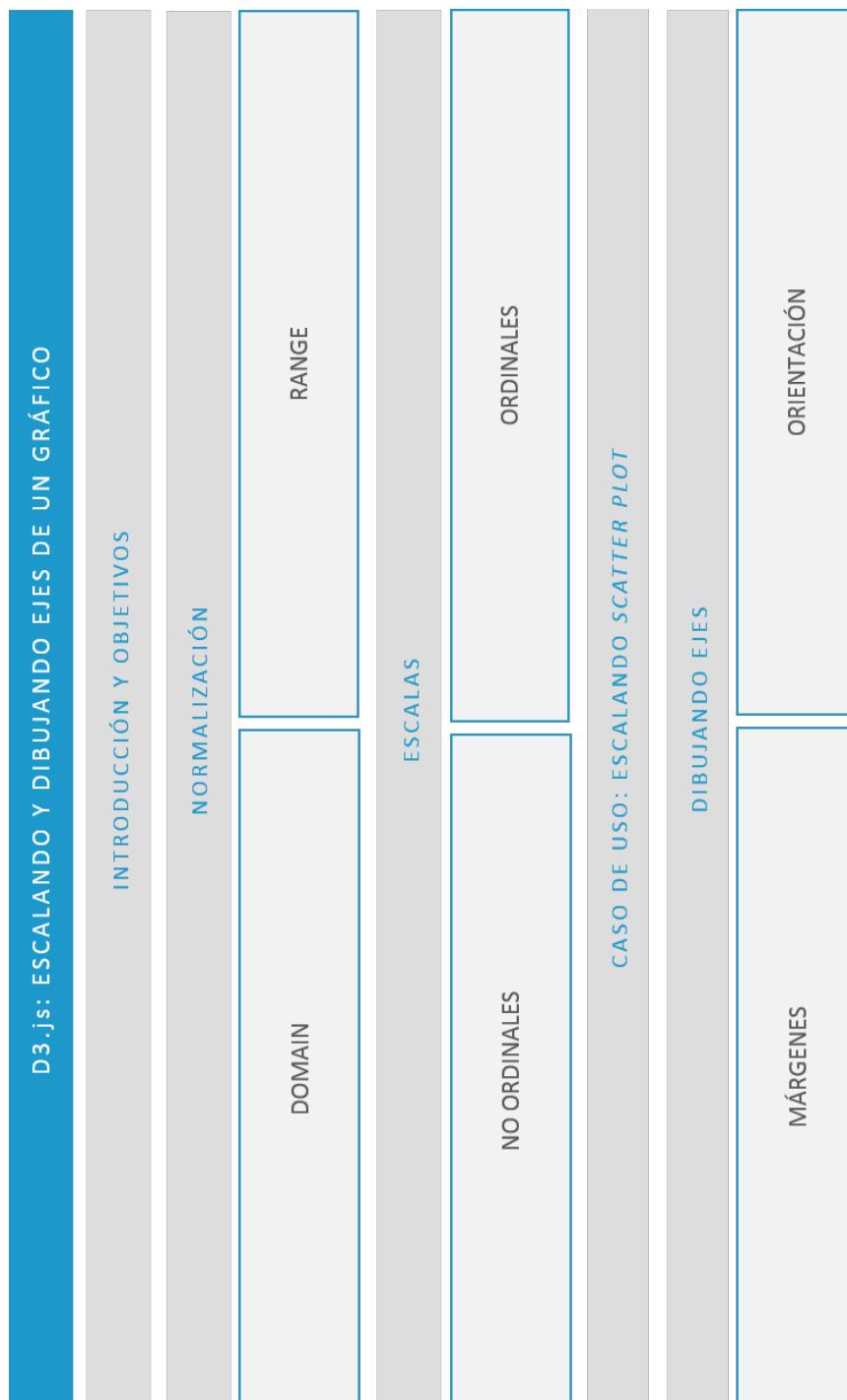
[Interactive Data Visualization for the Web](#)

[Setting up the Axes](#)

[Labelling multiple lines on a graph](#)

[How to rotate the text labels for the x Axis](#)

[Test](#)



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.

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>.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*.

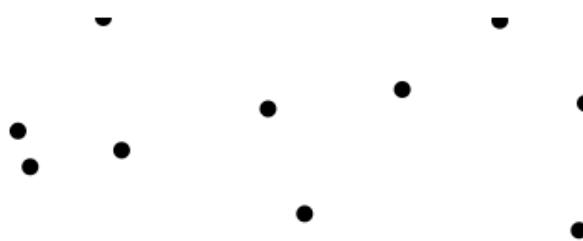


Figura 1. *Scatter plot* escalado a un rango definido.

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]);
```

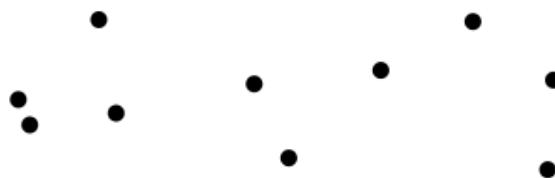


Figura 2. *Scatter Plot* escalado a un rango definido y con ajuste de márgenes.

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:

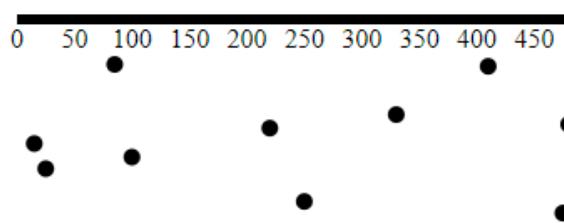


Figura 3. *Scatter Plot* con eje de la X en la parte superior.

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:

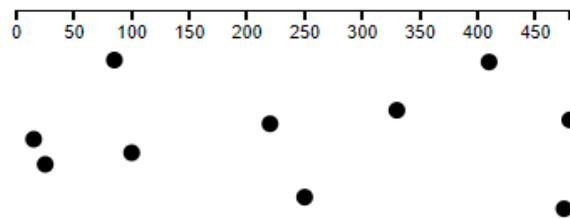


Figura 4. Scatter Plot y aplicación de estilo al eje de la X.

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.

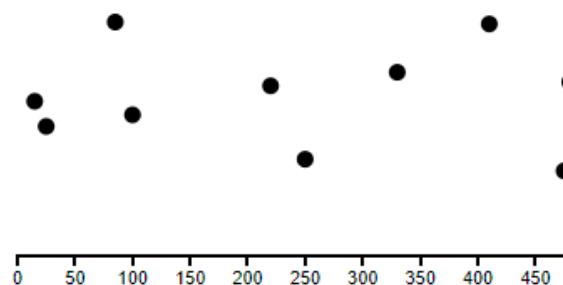


Figura 5. *Scatter Plot* con eje X situado en la parte inferior de la gráfica.

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:

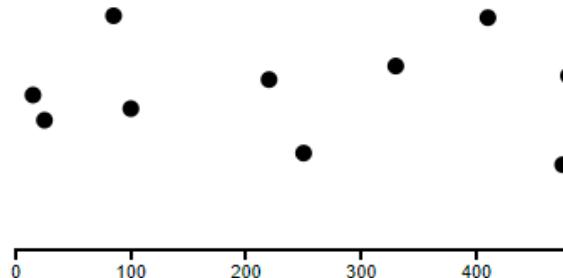


Figura 6. *Scatter Plot* con cinco divisiones.

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:

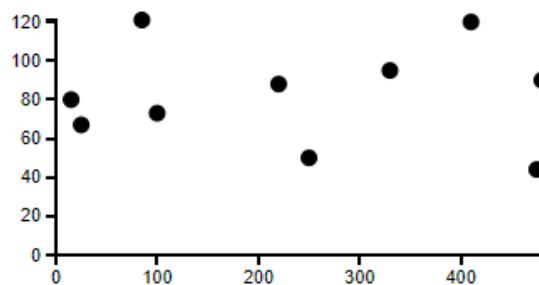


Figura 7. *Scatter Plot* con ejes X e Y.

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:

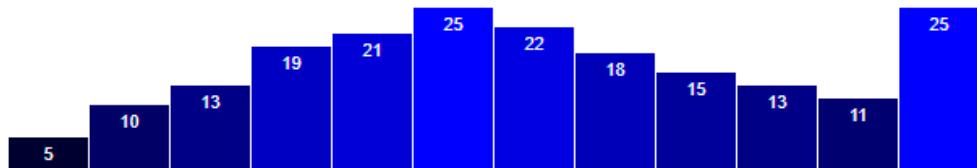


Figura 8. Bar Chart.

¿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:

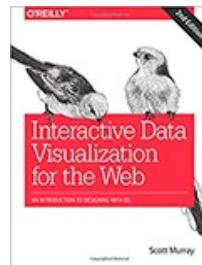
5.5. Referencias bibliográficas

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

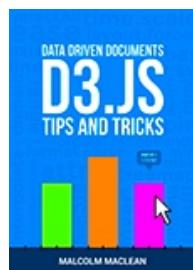
Murray, S. (2017). *Interactive Data Visualization for the Web*. [S. l.]: O'Reilly.



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. I.]: 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 d e <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.

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.

Herramientas de Visualización

Force Layout, transiciones, movimiento e interacción

Índice

[Esquema](#)

[A fondo](#)

Interactive Data Visualization for the Web

Anand S - Beautiful visualizations with d3.js

React y D3js trabajando juntos

Data Driven Delirious: An Intro to Data Visualisation Using D3.js

Dimple

Visual Sedimentation

NVD3

[Test](#)

[Ideas clave](#)

6.1. Introducción y objetivos

6.2. Force Layout

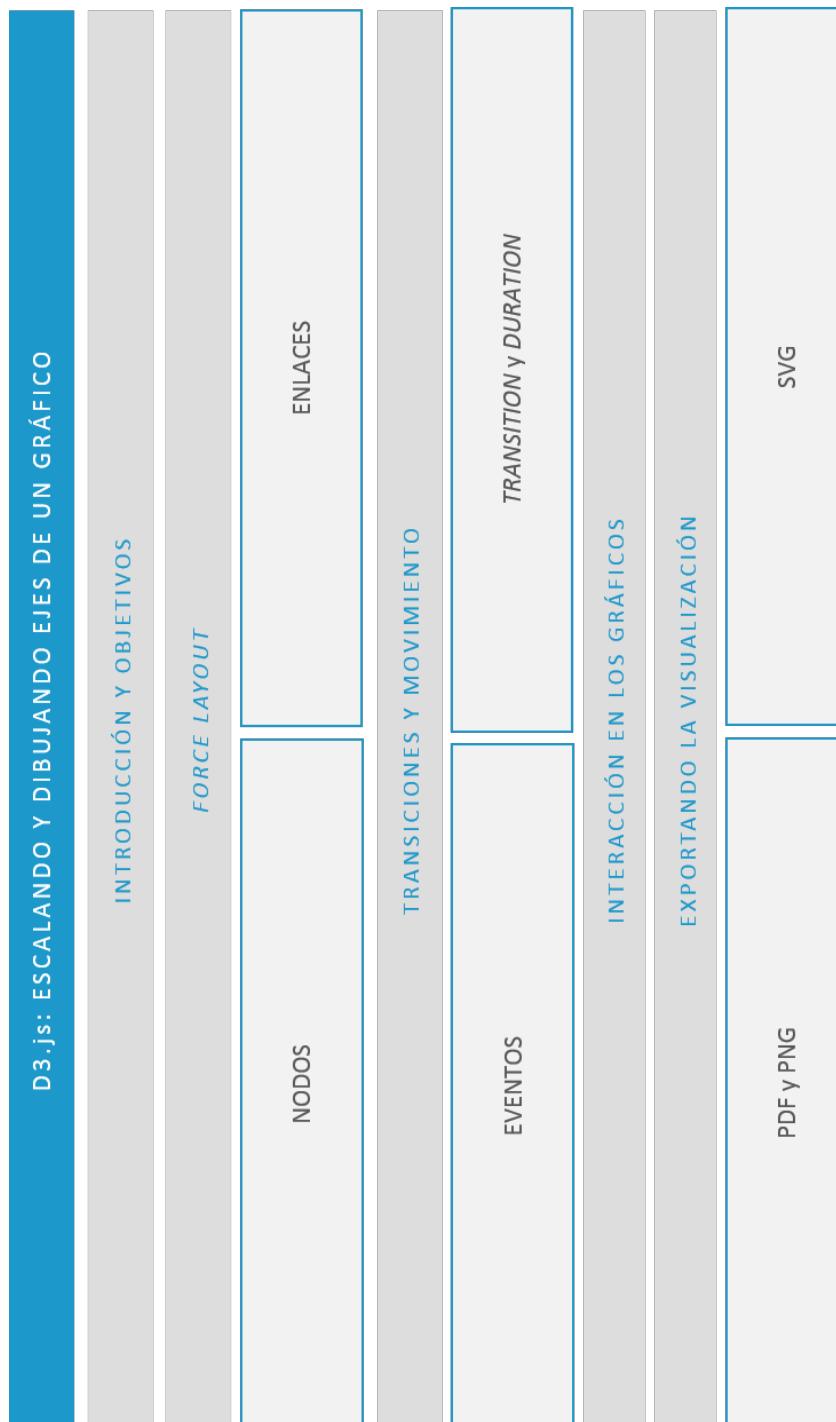
6.3. Actualización de gráficos con base en eventos

6.4. Transiciones y movimiento

6.5. Añadiendo interacción a los gráficos

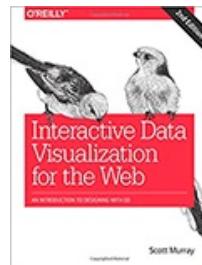
6.6. Exportando el resultado a PDF, Bitmaps y SVG

6.7. Referencias bibliográficas



Interactive Data Visualization for the Web

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

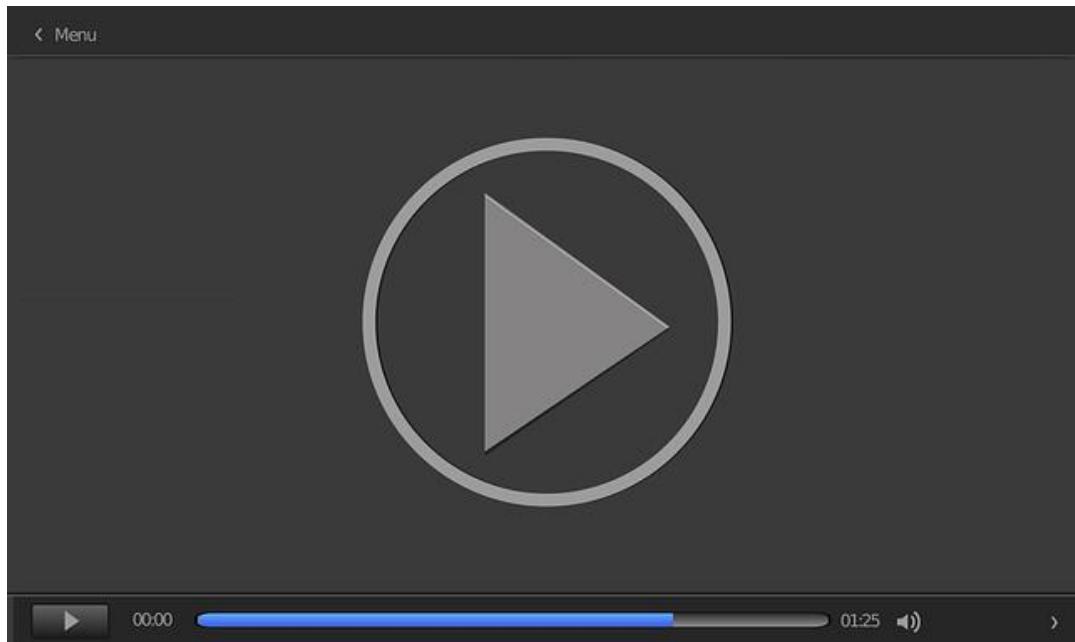


Murray, S. (2017). Dos son las secciones que nos interesan para este tema:
Transiciones y actualización de las escalas.

Anand S - Beautiful visualizations with d3.js

HasGeek TV. (6 de diciembre de 2012). *Anand S - Beautiful visualizations with d3.js* [Vídeo]. Youtube. <https://youtu.be/JAlZ0uJnqkA>

Esta interesante conferencia-taller trata los conceptos de la creación de visualizaciones con D3. Podrás sumergirte en los ejemplos específicos que muestran.



Accede al vídeo:

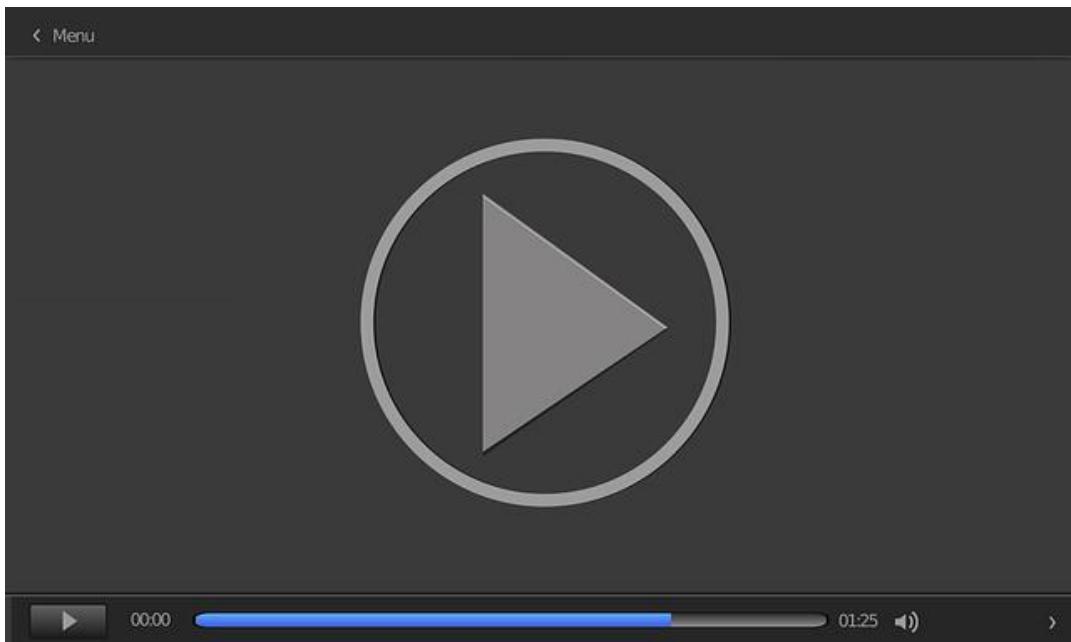
<https://www.youtube.com/embed/JAlZ0uJnqkA>

React y D3js trabajando juntos

Lemoncoders. (5 de julio de 2018). *React y D3js trabajando juntos* [Vídeo]. Youtube.

<https://youtu.be/8OcpV3tCBYU>

Siempre es interesante el uso de múltiples librerías combinándose entre sí. En este caso os mostramos el uso de D3.js y React.



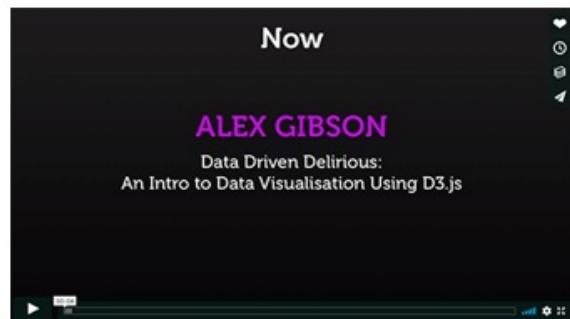
Accede al vídeo:

<https://www.youtube.com/embed/8OcpV3tCBYU>

Data Driven Delirious: An Intro to Data Visualisation Using D3.js

WDCNZ. (9 de julio de 2013). *Data Driven Delirious: An Intro to Data Visualisation Using D3.js* [Vídeo]. Vimeo. <https://vimeo.com/72467564>

Es bueno escuchar lo que nos pueden enseñar otros expertos, así que en este vídeo podéis encontrar una excelente charla de Alex Gibson en WDCNZ sobre visualización de datos con D3.js.



Dimple

Dimple. Página web oficial. <http://dimplejs.org/>

Mencionamos algunas librerías que, basadas en D3, intentan simplificar la toma de contacto con esta flexible librería. Con Dimple, convertiremos visualizaciones D3js en una librería bastante parecida en simplicidad a Google Charts.

dimple

An object-oriented API for business analytics
powered by d3.

Visual Sedimentation

Huron. S. (1 de abril de 2019). Visual Sedimentation.
Aviz. <https://www.aviz.fr/Research/VisualSedimentation>

Otra librería basada en D3.js, JQuery y Box2DWeb, se especializa en visualizar datos en tiempo real.



NVD3

NVD3. Página web oficial. <http://nvd3.org/>

Librería que simplifica de manera similar a Dimple las visualizaciones más populares basadas en D3.js.

NVD3 Re-usable charts for d3.js

6.1. Introducción y objetivos

D3 incorpora multitud de gráficas diferentes y, aunque cada una de ellas necesita parámetros específicos adaptados a sus particularidades, lo interesante es que dominéis aquellos principios básicos de D3 que son comunes a todas ellas.

En este tema abordaremos un nuevo tipo de visualización denominada *Force-Directed Graph* o *Force Layout*, orientada a representar las **relaciones entre entidades** mediante un grafo. Es una gráfica muy interesante y especialmente utilizada, por ejemplo, en el ámbito de las redes sociales para reflejar las relaciones existentes entre diferentes usuarios.

Ya sabemos cómo dibujar un gráfico por completo, en este tema también empezaremos a darle movimiento. D3 es una de las librerías de visualización más potentes, por lo que hemos de exprimirla al máximo.

Trataremos sobre cómo actualizar datos reaccionando a la generación de eventos y desarrollaremos una parte importante relativa al dinamismo de las visualizaciones, como son las transiciones y el movimiento de los objetos, para darle una percepción de realidad o para apoyar nuestro mensaje. Por último, veremos cómo se pueden exportar las visualizaciones.

6.2. Force Layout

En esta ocasión nos apoyaremos en uno de los *datasets* referenciados en el libro *Getting started with D3* de M. Dewar (2012). En concreto, utilizaremos uno de los *datasets* que el autor tiene en Github.

Accede al *dataset* desde el aula virtual o a través del siguiente enlace: https://github.com/mikedewar/getting_started_with_d3.git

Para generar este gráfico habría que partir del archivo .json, cargándolo por ejemplo en Brackets. Se puede descargar o copiar desde la siguiente dirección:

Accede al archivo .json desde el aula virtual o a través del siguiente enlace: https://github.com/mikedewar/getting_started_with_d3/blob/master/visualizations/data/stations_graph.json

Código de Force Layout

```
<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(){
      d3.json("stations_graph.json", function(json) {
        visualizeLayout(json);
      });
    }
    function visualizeLayout(data){
      var width = 1500
      var height = 1500
      var svg = d3.select("body")
        .append("svg")
```

```

.attr("width", width)
.attr("height", height);
var node = svg.selectAll("circle.node")
.data(data.nodes)
.enter()
.append("circle")
.attr("class", "node")
.attr("r", 12);
var link = svg.selectAll("line.link")
.data(data.links)
.enter().append("line")
.style("stroke","black");
var force = d3.layout.force()
.charge(-120)
.linkDistance(30)
.size([width, height])
.nodes(data.nodes)
.links(data.links)
.start();
force.on("tick", function() {
    link.attr("x1", function(d) { return d.source.x; })
    .attr("y1", function(d) { return d.source.y; })
    .attr("x2", function(d) { return d.target.x; })
    .attr("y2", function(d) { return d.target.y; });
    node.attr("cx", function(d) { return d.x; })
    .attr("cy", function(d) { return d.y; });
});
}
</script>
</head>
<body onload="read()">
</body>
</html>

```

Como siempre, dividiremos el código y lo explicaremos por partes. También explicaremos el archivo .json, que contiene los nodos (son paradas de metro) y las conexiones entre las paradas.

El gráfico que deberías obtener es el siguiente:

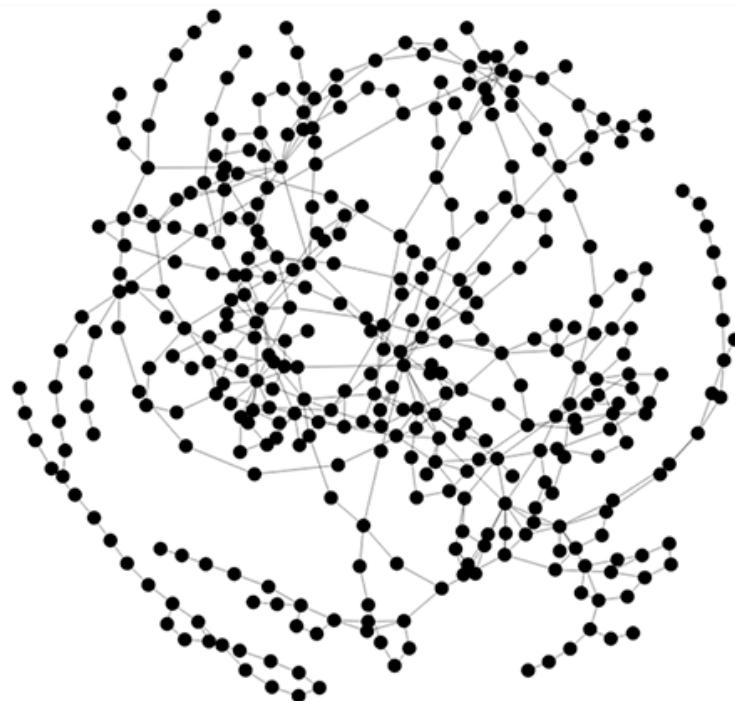


Figura 1. Gráfica con paradas de metro y las conexiones entre dichas paradas.

Fíjemonos en las siguientes líneas de código que definen los **círculos o puntos** del gráfico. Ahora sí que hay una diferencia en la identificación de los círculos, aunque por ahora, sea similar a cuando dibujamos el histograma.

```
var node = svg.selectAll("circle.node")
  .data(data.nodes)
  .enter()
  .append("circle")
  .attr("class", "node")
  .attr("r", 12);
```

Y ahora definimos las **líneas** entre los puntos.

```
var link = svg.selectAll("line.link")
  .data(data.links)
  .enter()
  .append("line")
  .style("stroke", "black");
```

A continuación, definimos el **algoritmo** que queremos utilizar. En este caso . Y para este algoritmo en concreto, se utilizan los atributos `x` y `y`.

```
var force = d3.layout.force()  
    .charge(-120)  
    .linkDistance(30)  
    .size([width, height])  
    .nodes(data.nodes)  
    .links(data.links)  
    .start();
```

Con la función `adaptamos` adaptamos los valores de nuestro *dataset* a los requerimientos del algoritmo:

```
force.on("tick", function() {  
    link.attr("x1", function(d) { return d.source.x; })  
    .attr("y1", function(d) { return d.source.y; })  
    .attr("x2", function(d) { return d.target.x; })  
    .attr("y2", function(d) { return d.target.y; });  
    node.attr("cx", function(d) { return d.x; })  
    .attr("cy", function(d) { return d.y; });
```

Por último, algo muy típico de este tipo de gráficos es proporcionar la habilidad de arrastrar los nodos por la pantalla. Si añadimos línea de código, lograremos arrastrar cualquier nodo con solo hacer clic en él.

6.3. Actualización de gráficos con base en eventos

Como hemos explorado en otras librerías, todas las visualizaciones tienen sus eventos. ¿Cómo podemos generar esa interacción en nuestras propias visualizaciones? Con D3 veremos que todas estas acciones son muy fáciles. Hemos de entender cómo se producen, pero una vez que entendamos los conceptos básicos, no serán más que unas cuantas líneas de código.

«*Time has been transformed, and we have changed; it has advanced and set us in motion; it has unveiled its face, inspiring us with bewilderment and exhilaration*» (Brainy Quote).

El movimiento es el secreto de la vida. No importa lo mal que estemos, debemos seguir hacia delante pase lo que pase. Las visualizaciones son reflejo de la vida. Dan mensajes, expresan ideas, sugieren caminos, por lo que darles movimiento parece ser una idea coherente a estos principios. Este tema trata de esto: dar interacción y movimiento a nuestras visualizaciones.

Tomemos como base para continuar el código de la gráfica *bar chart* con escala ordinal que desarrollamos en el anterior tema. Sobre dicho código vamos a explicar cómo trabajar con eventos.

Vamos a seguir cuatro pasos:

- ▶ Crear un botón que genere un evento para actualizar nuestro gráfico.
- ▶ Crear un nuevo *dataset* y actualizar la gráfica con él.
- ▶ Actualizar las dimensiones del gráfico.
- ▶ Hacer efectiva las actualizaciones.

Crear un botón

Todo elemento en D3.js puede ser un botón, siendo más estrictos, **todo elemento puede generar eventos**. Para comenzar, generaremos un texto y le asignaremos un *pop-up* como resultado a nuestro evento.

```
d3.select("body")
.append("p")
.text("Pulsa aquí")
.on("click", function() {
    alert("Conseguido! Ahora vamos a hacer algo útil...");
});
```

Con este código antes de la definición del svg, al pinchar en el texto de «Pulsa aquí», se obtiene el siguiente resultado:



Figura 2. Evento en una gráfica.

Generar un nuevo *dataset*

Esto nos va a servir para actualizar en tiempo real la gráfica al pulsar el botón, que es nuestro objetivo real. El nuevo *dataset* sería el siguiente:

Actualizar las dimensiones del gráfico

El siguiente paso sería actualizar los valores de nuestro gráfico y adaptar altura y anchura al nuevo *dataset*:

```
svg.selectAll("rect")
  .data(dataset)
  .attr("y", function(d) {
    return h - yScale(d);
})
  .attr("height", function(d) {
    return yScale(d);
});
```

Introducimos todo esto en nuestro botón:

```
d3.select("body")
  .append("p")
  .text("Pulsa aquí")
  .on("click", function() {
    dataset = [ 11, 12, 15, 20, 18, 17, 16, 18, 23, 25, 5, 10];
    svg.selectAll("rect")
      .data(dataset)
      .attr("y", function(d) {
        return h - yScale(d);
})
      .attr("height", function(d) {
        return yScale(d);
});
  });
});
```

Obtendremos el siguiente resultado:

Pulsa aquí

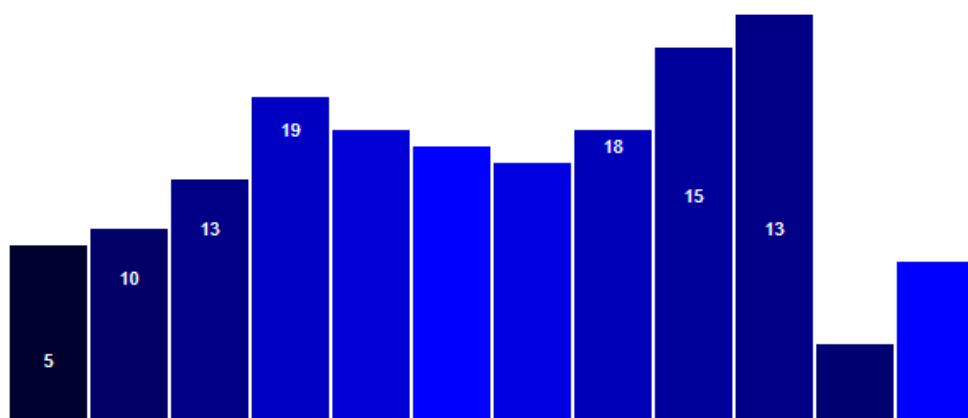


Figura 3. Gráfica actualizada con un nuevo *dataset*.

Hacer efectiva las actualizaciones

Si miramos el gráfico, vemos que no hemos actualizado el texto. Añadamos al evento el siguiente código:

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

Además, vamos a hacer que después de pulsar en el texto, este desaparezca borrando nuestro botón. Para ello añadimos lo siguiente:

Este es el resultado final:

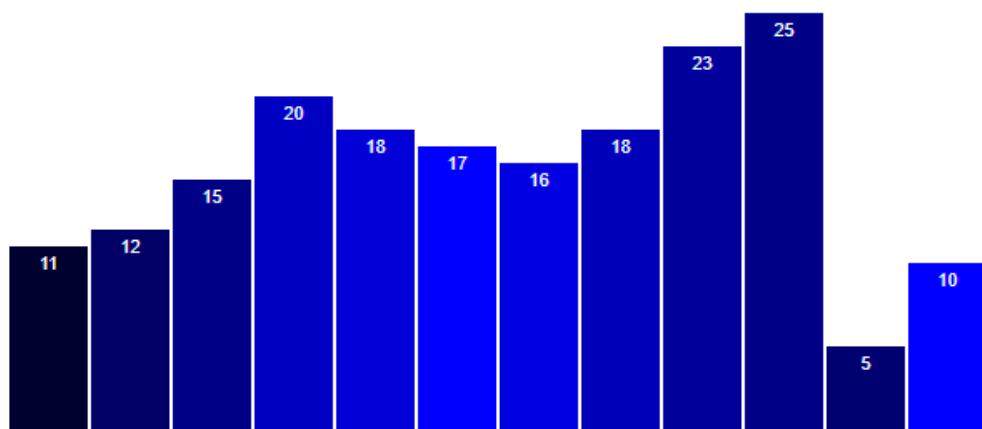


Figura 4. Gráfica final con el *dataset* actualizado.

6.4. Transiciones y movimiento

Las transiciones son el movimiento que dará vida a nuestro gráfico y en D3.js podemos hacerlo con una simple línea de código. Existe división de opiniones en cuanto a las transiciones: para algunos, estas atraen a usuarios más noveles y otros las consideran un adorno innecesario en visualizaciones.

¿Cómo se hacen las transiciones? Una simple línea de código: **transition()**. Se incluye en el código donde creábamos el botón para actualizar el *bar chart*. Al pulsar sobre tu evento, verás que el gráfico será animado.

```
svg.selectAll("rect")
  .data(dataset)
  .transition()
  .attr("y", function(d) {
    return h - yScale(d);
})
  .attr("height", function(d) {
    return yScale(d);
});
```

¿Qué podemos controlar en estas transiciones? Una de las cosas que podemos controlar es la **duración** utilizando duration(milisegundos). Probad este código:

```
svg.selectAll("rect")
  .data(dataset)
  .transition()
  .duration(3000)
  .attr("y", function(d) {
    return h - yScale(d);
})
  .attr("height", function(d) {
    return yScale(d);
});
```

hace que la animación dure tres segundos. Verás que las barras «crecen», sin

embargo, el texto de las barras está fijo. Por tanto, también deberíamos aplicar la **transición al texto**.

```
svg.selectAll("text")
  .data(dataset)
  .transition()
  .duration(3000)
  .text(function(d) {
    return d;
})
  .attr("x", function(d, i) {
    return xScale(i) + xScale.rangeBand() / 2;
})
  .attr("y", function(d) {
    return h - yScale(d) + 14;
});
```

Si nos fijamos en la transición, primero es lenta y luego acelera. Esto lo podemos modificar y regular con otra función () y hacer la **transición lineal**.

```
svg.selectAll("rect")
  .data(dataset)
  .transition()
  .duration(3000)
  .ease("linear")
  .attr("y", function(d) {
    return h - yScale(d);
})
  .attr("height", function(d) {
    return yScale(d);
});
```

Si solo modificas las barras pero no el texto, notarás la diferencia de cómo evolucionan los dos elementos. Prueba con otros métodos, no solo el lineal. Por ejemplo, o y observa cómo reacciona.

Otra de las particularidades que tenemos a nuestra disposición consiste en **retrasar la animación** con la función `.delay()`. Retrasaremos tres segundos nuestra animación:

```
svg.selectAll("rect")
  .data(dataset)
  .transition()
  .duration(3000)
    .ease("linear")
  .delay(3000)
  .attr("y", function(d) {
    return h - yScale(d);
})
  .attr("height", function(d) {
    return yScale(d);
});
```

6.5. Añadiendo interacción a los gráficos

Los modos de interaccionar con un gráfico son tan diversos como los estilos de los diseñadores. Por ejemplo, imaginemos que queremos que reaccione el valor al pulsar una barra. Al saber el valor, también podemos hacer otras acciones. Practiquemos cómo ver el valor, aunque esté escrito en la barra, en una ventana abierta.

```
svg.selectAll("rect")
  .data(dataset)
  .on("click", function(d) {
    alert(d);
})
  .transition()
  .ease("linear")
.attr("y", function(d) {
  return h - yScale(d);
})
.attr("height", function(d) {
  return yScale(d);
});
```

Haz clic en cualquiera de las barras y aparecerá una pequeña ventana con el valor de dicha barra.

Ahora bien, si queremos hacer algo más interesante, por ejemplo, hacer cambiar las barras cuando pasamos por encima, podemos probar el siguiente código:

```
svg.selectAll("rect")
  .data(dataset)
  .on("mouseover", function(d) {
d3.select(this)
  .style("fill", "red");
})
  .on("mouseout", function(){
d3.select(this)
```

```
.style("fill", function(d) {
    return "rgb(0, 0, " + (d * 10) + ")";
})
.transition()
.ease("linear")
.attr("y", function(d) {
    return h - yScale(d);
})
.attr("height", function(d) {
    return yScale(d);
});
```

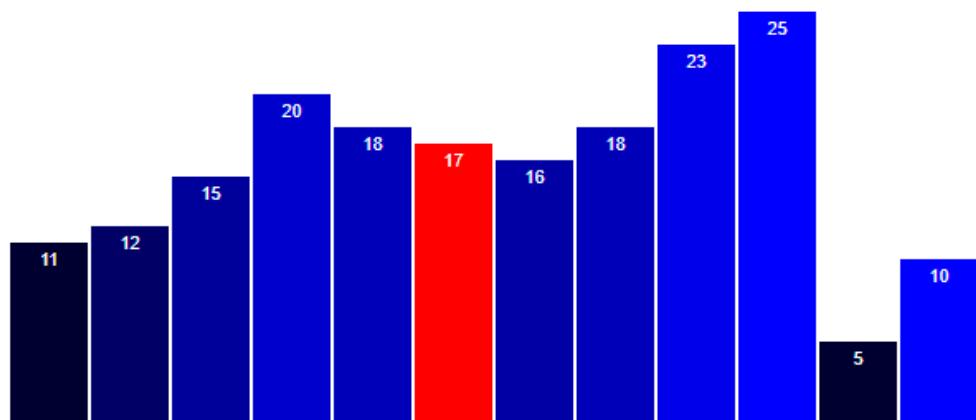


Figura 5. Gráfica con efecto de barra resaltada.

6.6. Exportando el resultado a PDF, Bitmaps y SVG

Esta sección no está muy relacionada con D3.js en sí, pero es evidente la utilidad de poder exportar nuestro gráfico desde la ventana del navegador a otros formatos.

- ▶ La primera opción es la captura de pantalla (Print Screen en PC y Command ⌘ + Shift + 3 en MAC), que nos generará un archivo de tipo Bitmap o PNG. Es la forma más tradicional y fácil para obtener resultados, aunque de baja calidad.
- ▶ También podemos imprimir en PDF, seleccionando dicho formato en la pantalla de opciones de impresión.
- ▶ La opción más interesante es trabajar con **SVG**, así podremos modificar la imagen en Photoshop, escalar la imagen a todos los tamaños, etc.

Para eso, en el caso anterior, seleccionaremos el gráfico, pulsaremos el botón derecho del ratón y seleccionaremos **Inspeccionar**, como indica la siguiente figura:

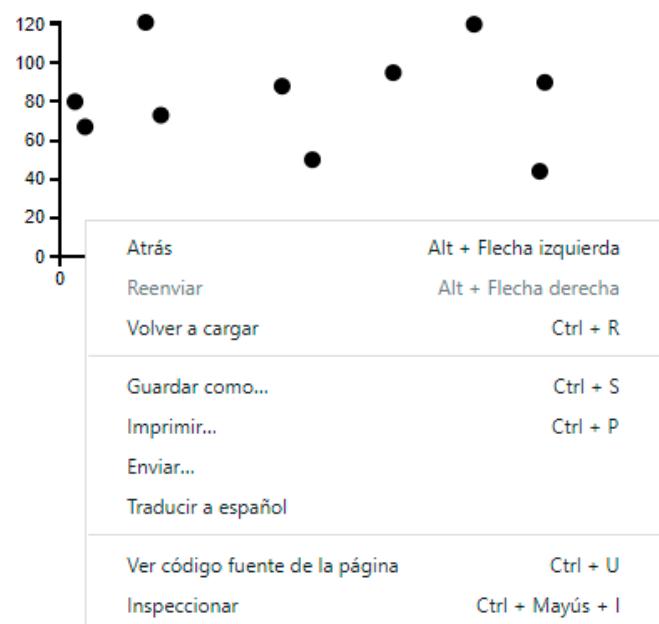


Figura 6. Inspeccionar elemento.

Seleccionaremos todo lo que hay entre las etiquetas <svg>...</svg>.

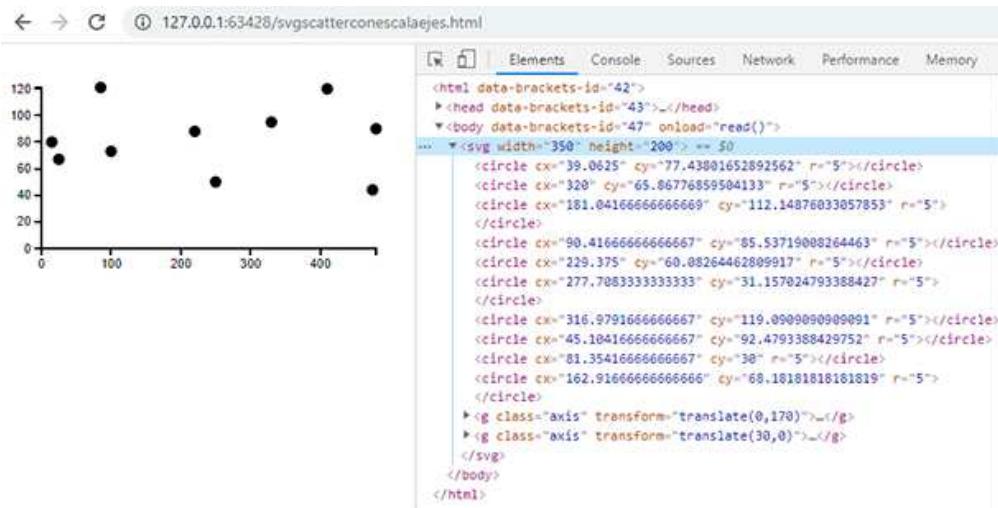


Figura 7. Detalle código SVG.

Lo copiaremos en un archivo nuevo que tenga la extensión .svg. Este archivo lo podrás abrir con Acrobat Reader, Photoshop, etc.

6.7. Referencias bibliográficas

Brainy

Quote. Khalil

Gibran

Quotes.

https://www.brainyquote.com/quotes/khalil_gibran_101832

Dewar, M. (2012). *Getting Started with D3*. (S. l.): O'Reilly Media.

1. ¿Cuánto tardará la transición en este código?

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

- A. 1 seg.
- B. 2 seg.
- C. Depende del número de barras del gráfico.
- D. No hay transición.

2. ¿Cuánto tardará la transición en este código?

```
svg.selectAll("text")
  .data(dataset)
  .text(function(d) {
    return d;
  })
  .transition()
  .delay(1000)
  .attr("x", function(d, i) {
    return xScale(i) + xScale.rangeBand() / 2;
  })
  .attr("y", function(d) {
    return h - yScale(d) + 14;
  });

```

- A. 1 seg.
- B. 2 seg.
- C. Depende del número de barras del gráfico.
- D. El tiempo por defecto transición.

3. ¿Es la sintaxis correcta?

```
on("mouseover", function(d) {
  d3.select(this)
    .style("fill", "red");
})
```

- A. Sí.
- B. No, porque falta la selección del objeto.
- C. No, porque el evento mouseover no existe.
- D. No, porque la llamada a la función es incorrecta.

4. ¿Es la sintaxis correcta?

```
on("outmouse", function(d) {  
d3.select(this)  
.style("fill", "red");  
})
```

- A. Sí.
- B. No, porque falta la selección del objeto.
- C. No, porque el evento outmouse no existe.
- D. No, porque la llamada a la función es incorrecta.

5. ¿Podemos escuchar eventos en los elementos p de D3?

- A. Sí.
- B. No, porque hacen falta objetos que puedan aceptar un clic de ratón sobre ellos.
- C. No, porque los eventos aplican solo a los datos.
- D. No, porque p no es un elemento de D3.

6. ¿Podemos escuchar/generar eventos en los elementos rect de D3?

- A. Sí.
- B. No, porque hacen falta objetos que puedan aceptar un clic de ratón sobre ellos.
- C. No, porque los eventos aplican solo a los datos.
- D. No, porque rect no es un elemento de D3.

7. ¿Qué conseguimos cuando utilizamos en la transición la función ?

 - A. Que la transición sea constante.
 - B. Que la transición dure lo mínimo posible.
 - C. Que la transición de los objetos sea vertical.
 - D. Que la transición de los objetos sea horizontal.
8. Si queremos exportar nuestra visualización a un archivo svg, tenemos que:

 - A. Exportar nuestro gráfico con la función D3.exportSVG.
 - B. Copiar el código HTML generado y copiarlo en un archivo svg.
 - C. No existe la posibilidad. Se pueden crear y visualizar gráficos SVG pero no exportarlos.
 - D. Ninguna de las anteriores.
9. Utilizando captura de pantalla para exportar nuestra visualización a otro formato:

 - A. Generamos un archivo Bitmaps.
 - B. Generamos un archivo SVG.
 - C. Generamos un archivo PDF.
 - D. No es posible capturar la pantalla.
10. ¿Cuál es el comportamiento al aplicar la función ?

 - A. Los nodos se pueden arrastrar y, cuando sueltas el nodo, este se queda en esa misma posición.
 - B. Los nodos se pueden arrastrar y, cuando sueltas el nodo, vuelven a su posición original.
 - C. Los nodos se mueven de forma independiente por sí mismos en un ciclo continuo.
 - D. Ninguna de las anteriores.

Herramientas de Visualización

Tema 7. Power BI

Índice

[Esquema](#)

[Ideas clave](#)

[7.1. Introducción y objetivos](#)

[7.2. Instalación e Interfaz de Power BI](#)

[7.3. Funciones de Power BI](#)

[7.4. Obtención, preparación y modelado de datos](#)

[7.5. Visualización de datos](#)

[7.6. Visualizaciones avanzadas](#)

[A fondo](#)

[Introducing Microsoft Power BI](#)

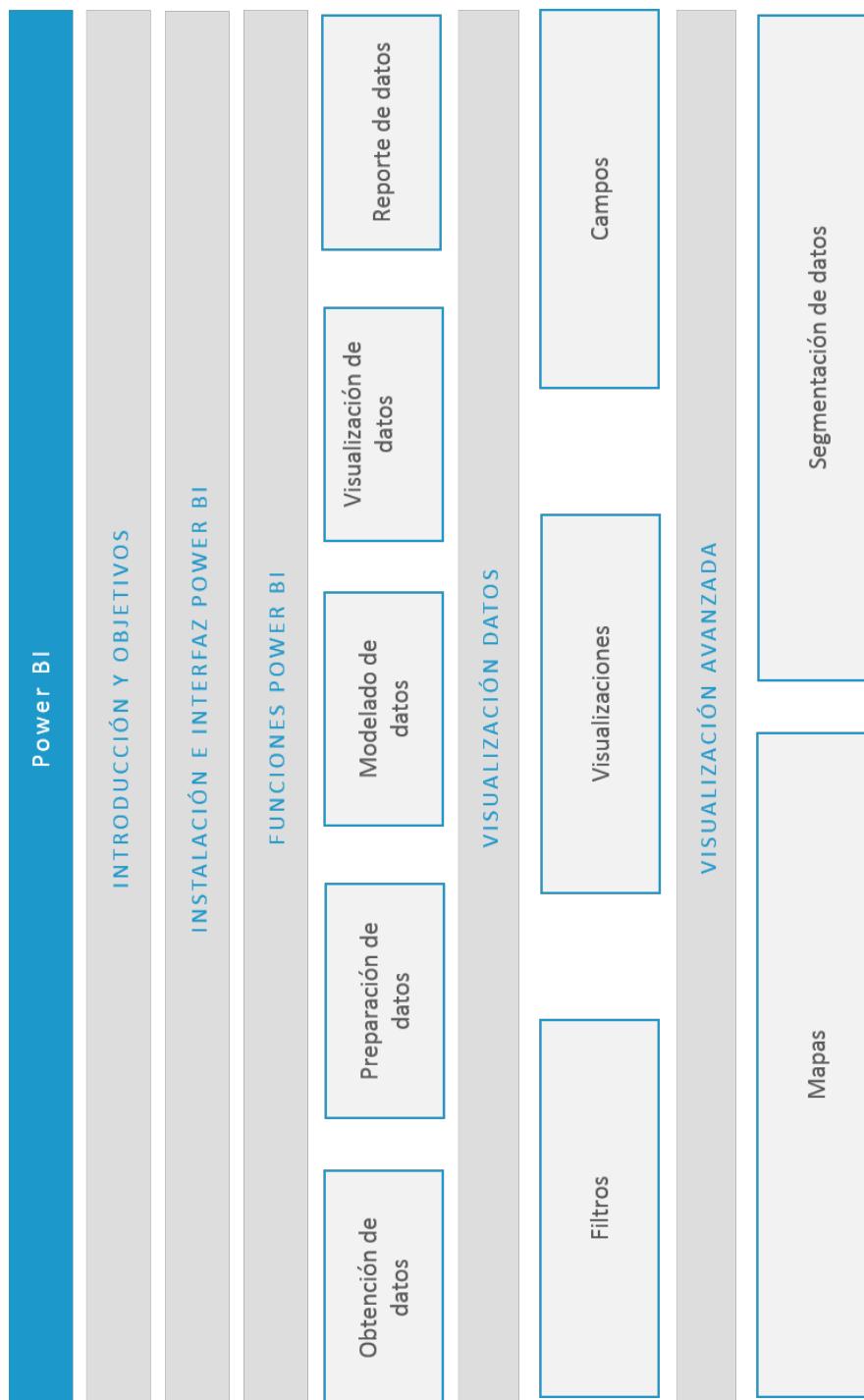
[Applied Microsoft Power BI: Bring your data to life!](#)

[Making Cities Smarter: How Citizens' Collective Intelligence Can Guide Better Decision Making](#)

[Microsoft Power BI Community](#)

[Test](#)

Esquema



7.1. Introducción y objetivos

Power BI puede integrar datos de múltiples y relacionarlos para obtener información relevante y así, poder tomar decisiones desde cualquier dispositivo. Aprenderemos con los fundamentos de cómo utilizar Power BI, sus principales funciones y cómo desarrollar proyectos de visualización con esta herramienta de Microsoft.

Microsoft Power BI está situado como uno de los productos líderes de *Business Intelligence*, según el «cuadrante mágico de Gartner», junto a Tableau y Qlik. Con esta herramienta podrás transformar millones de datos, relacionar múltiples fuentes, realizar cálculos complejos e interactuar fluidamente con visualizaciones. Su interfaz y diseño es muy amigable e intuitivo y cualquier persona puede utilizarla sin conocimientos técnicos avanzados.



Figura 1. Cuadrante mágico de Gartner sobre plataformas de analítica e inteligencia de negocio. Fuente: <https://www.qlik.com/us/gartner-magic-quadrant-business-intelligence>

Durante años, después de su lanzamiento en 2013, Power BI fue un producto «seguidor» que solo tenía que ser «lo suficientemente bueno», dado su política de

precios bajos. Sin embargo, en la actualidad ha superado a la mayoría de sus competidores en términos de funcionalidad.

Accede a la página oficial de Power Bi desde el siguiente enlace: <https://powerbi.microsoft.com/es-es/>

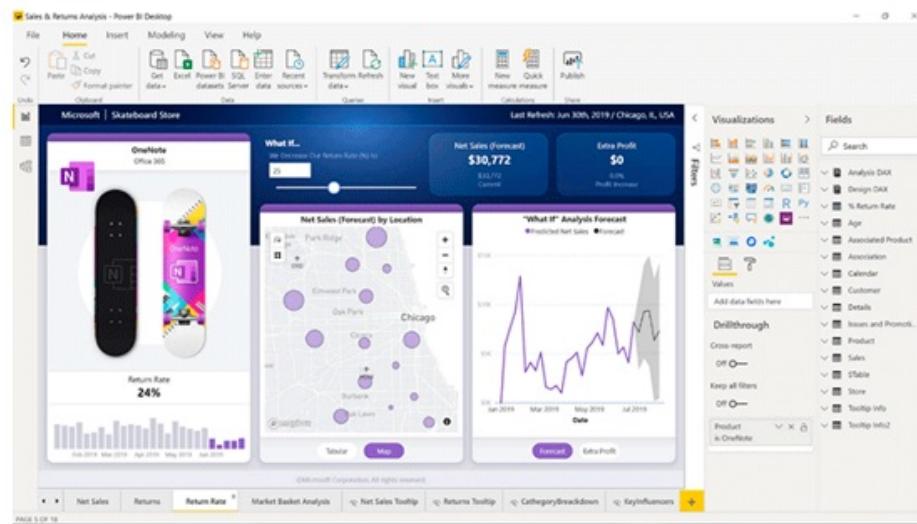


Figura 2. Microsoft Power BI. Fuente: <https://powerbi.microsoft.com/es-es/desktop/>

7.2. Instalación e Interfaz de Power BI

Power BI está disponible como una opción SaaS, que se ejecuta en la nube de Azure o como una opción local a través de su Servidor de informes Power BI. También se puede utilizar Power BI Desktop como una herramienta de análisis personal gratuita e independiente.

Accede a Power BI Desktop desde el siguiente enlace: <https://powerbi.microsoft.com/es-es/desktop/>

El proceso de instalación es prácticamente automático y solo nos pedirá un *e-mail* para registrar la herramienta.

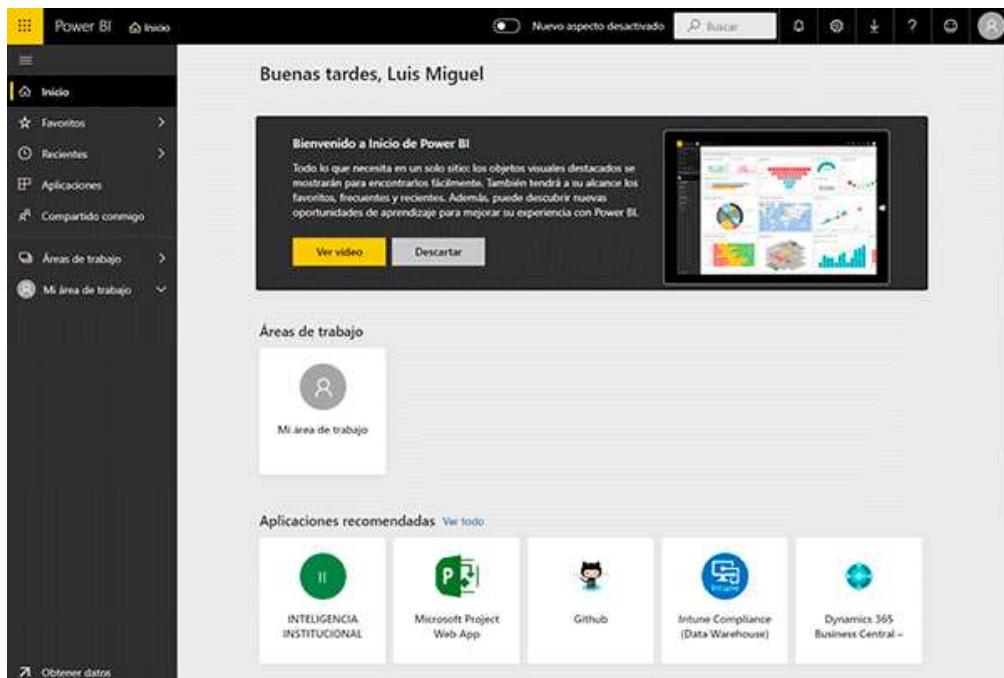


Figura 3. Página de bienvenida de Power BI.

Respecto a la **interfaz de Power BI**, esta presenta un aspecto muy similar a todos

los productos de Microsoft Office.

Dispone de una **barra de herramientas superior** donde podemos encontrar opciones de menú generales como Archivo, que permite las operaciones habituales de Abrir, Guardar, Importar/Exportar informes, Exportar la visualización a PDF, entre otras muchas opciones.

El resto de opciones del menú, tales como Inicio, Insertar y Modelado, incorporan elementos específicos que nos permitirán crear nuestras visualizaciones, aunque en su mayor parte se pueden emplear a través de las opciones de los menús laterales.

Existe un **menú lateral izquierdo**, a través del cual podremos acceder a diferentes vistas de los informes, pasando de la visualización de las gráficas a la visualización de los datos.

En la **parte inferior de la pantalla**, existe la opción de agregar nuevas solapas a la visualización, como si fueran pestañas de una hoja Excel.

En el **menú lateral derecho** están dispuestas las opciones de Filtros, Visualizaciones y Campos. Estas opciones nos permiten crear y aplicar diferentes filtros a los datos y, en consecuencia, influir en los criterios de visualización. También podremos escoger con un solo clic el tipo de gráfica que queremos incorporar a nuestro proyecto y seleccionar los campos que queremos incluir en él.

Iremos profundizando en estas opciones a largo de los diferentes casos de uso que vamos a desarrollar.

Ideas clave

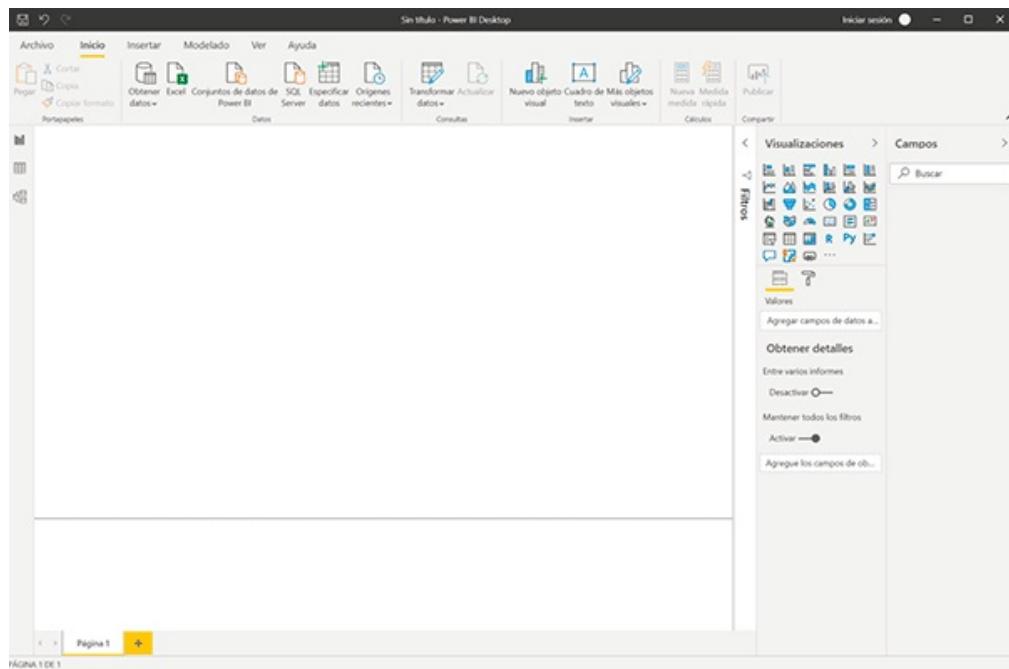


Figura 4. Interfaz Power BI.

7.3. Funciones de Power BI

Todo proyecto de análisis y visualización de datos con Power BI puede estructurarse en los siguientes apartados funcionales:

Obtención de datos: cargar datos desde diferentes fuentes de datos como archivos de texto planos, hojas Excel, entornos relacionales como SQL Server o servicios en línea como Facebook o Google Analytics.

La capacidad de Power BI para integrarse con diferentes fuentes de datos es muy amplia.

Preparación de datos: limpiar y tratar los datos de una manera estructurada y, para ello, podremos emplear el gestor de datos integrado en Power BI denominado Query Manager.

Modelado de datos: relacionar datos procedentes de diferentes fuentes y realizar cálculos que nos permitan descubrir información relevante, ajustando nuestros datos y dándoles una estructura que nos facilite su posterior visualización.

Visualización de datos: dotar a los usuarios de gráficos, tablas y representaciones que les permitan analizar los datos para tomar decisiones.

Reporte de datos: crear cuadros de mandos o *dashboards* que mejoren la comunicación de los informes tanto desde un punto de vista visual como de usabilidad, creando vistas, aplicando filtros, etc.

A continuación, comenzaremos a desarrollar un proyecto completo en Power BI de análisis y visualización de datos que nos permite recorrer todos estos bloques. En este proyecto hablaremos de Smart Cities y de cómo se pueden utilizar los datos para que las ciudades sean cada vez más inteligentes.

[London Data Store](#)

Londres fue una de las primeras ciudades en Europa que apostó por las iniciativas Open Data o filosofía de datos abiertos, defendiendo que determinada información generada por las propias ciudades debería ser accesible y utilizable por el público en general, sin requerir permisos específicos.

El Data Store de Londres contiene más de 500 *datasets* abiertos con datos relativos a vivienda, transporte, medio ambiente, etc.

Accede a la página web de Data Store de Londres: <https://data.london.gov.uk/>

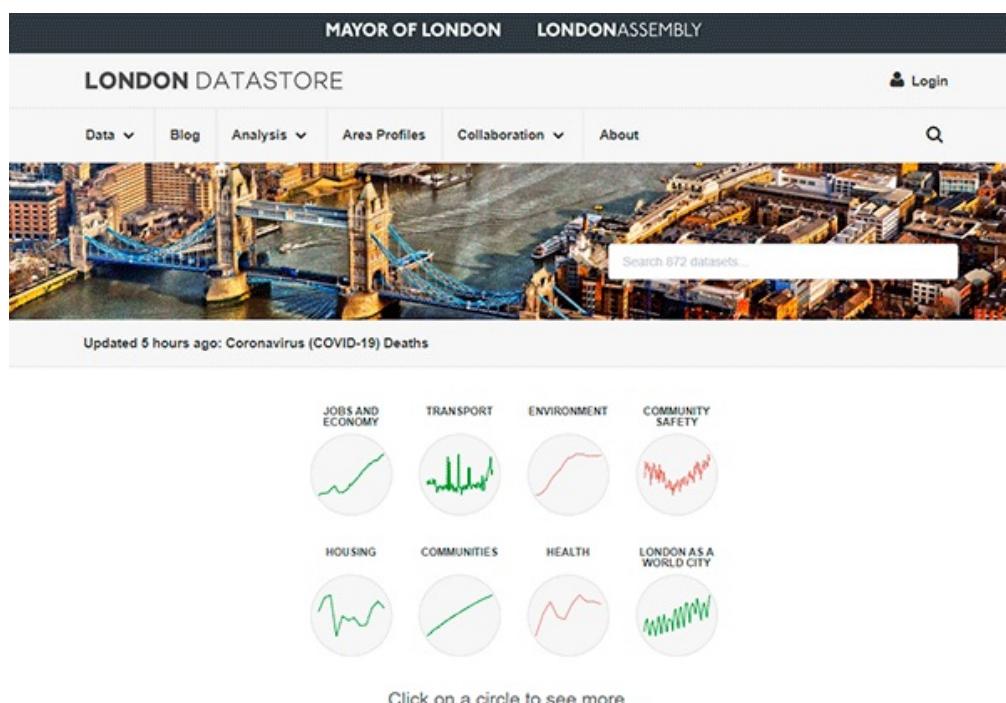


Figura 5. *London Data Store*. Fuente: <https://data.london.gov.uk/>

El *dataset* concreto que vamos a emplear es muy pintoresco. Recopila toda la información disponible sobre los servicios de rescate animal del cuerpo de bomberos de Londres (*London Fire Brigade*, LFB).

Accede a la página web del LFB: <https://data.london.gov.uk/dataset/animal-rescue-incidents-attended-by-lfb>

The screenshot shows a data visualization interface for the 'Animal rescue incidents attended by LFB' dataset. At the top, there is a logo for 'LFB LONDON FIRE BRIGADE' and a title 'Animal rescue incidents attended by LFB'. Below the title, it says 'London Fire Brigade' and 'Data'. A note indicates the data was created 4 years ago and updated a month ago. The main content area contains several paragraphs of descriptive text about the data, mentioning non-fire incidents, assistance to animals, monthly updates, location information, and cost calculations. It also notes that the London Fire Commissioner is the authority for London. Below the text, there is a navigation bar with categories: LFB London Fire Brigade, LFB, Emergency response, rescue, special services, animals, and animal. Underneath the navigation bar, there is a preview section for the dataset, showing a table with columns for 'Ward' and other data. At the bottom, there is an 'Author' section with an email icon and the name 'LFB Information Management'.

Figura 6. Descarga de datos de trabajo del cuerpo de bomberos de Londres (London Fire Brigade). Fuente: <https://data.london.gov.uk/dataset/animal-rescue-incidents-attended-by-lfb>

Se trata de una tabla con más de 6500 registros, con la información de las salidas registradas de los bomberos para prestar el servicio de rescate de animales. Los campos básicos de la tabla son el indicador de la salida, la fecha del aviso, el tiempo dedicado, el coste incurrido en el servicio, el tipo de animal, origen de la llamada, localización, etc.

7.4. Obtención, preparación y modelado de datos

En este caso vamos a trabajar con el fichero que vamos a descargar, «Animal Rescue incidents attended by LFB from Jan 2009.csv». El primer paso, por lo tanto, es importar dicho fichero en Power BI: **Obtener datos > Excel**.

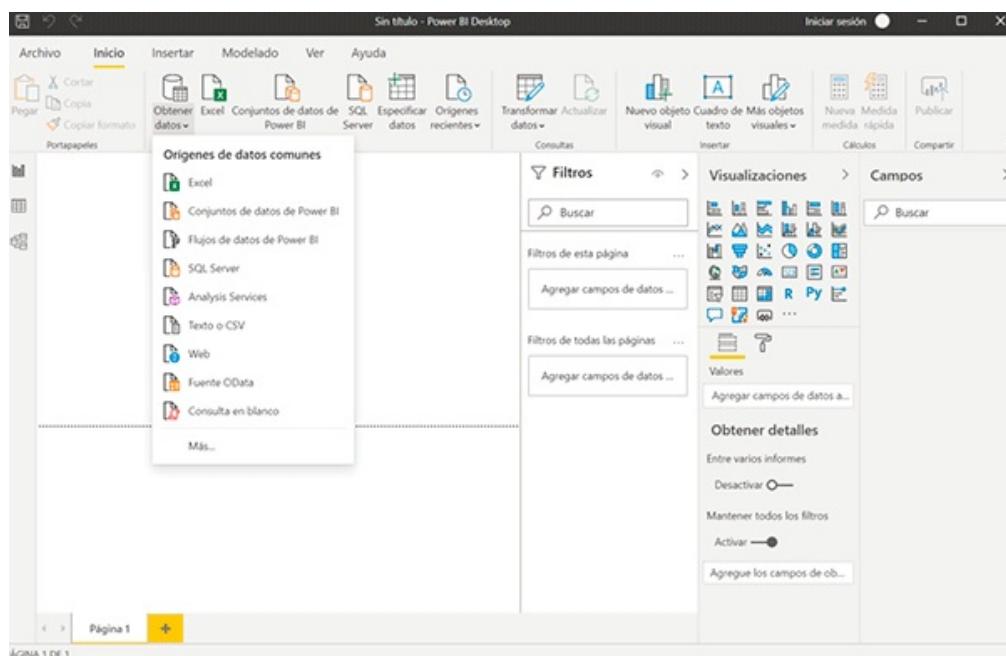


Figura 7. Carga de datos.

La pantalla de selección nos da la opción de seleccionar las hojas de Excel que queremos cargar y, a la vez, nos ofrece una visualización parcial de los datos.

En este momento podemos utilizar la opción de editar para hacer todos los cambios de formato que sean necesarios.

Animal Rescue incidents attended by LFB from Jan 2009.csv

Origen de archivo	Delimitador	Detección del tipo de datos						
1252: Europeo occidental (Windows)	Coma	Basado en las primeras 200 filas						
IncidentNumber	DateTimeOfCall	CalYear	FinYear	TypeOfIncident	PumpCount	PumpHoursTotal	HourlyNotionalCost[£]	Ind
239091	01/01/2009 3:01:00	2009	01/09/2008	Special Service	2	2	255	
275091	01/01/2009 8:51:00	2009	01/09/2008	Special Service	2	2	255	
2075091	04/01/2009 10:07:00	2009	01/09/2008	Special Service	2	2	255	
2872091	05/01/2009 12:27:00	2009	01/09/2008	Special Service	2	2	255	
3538091	06/01/2009 15:23:00	2009	01/09/2008	Special Service	2	2	255	
3742091	06/01/2009 19:30:00	2009	01/09/2008	Special Service	2	2	255	
4011091	07/01/2009 6:29:00	2009	01/09/2008	Special Service	2	2	255	
4711091	07/01/2009 11:55:00	2009	01/09/2008	Special Service	2	2	255	
4306091	07/01/2009 13:48:00	2009	01/09/2008	Special Service	2	2	255	
4715091	07/01/2009 21:24:00	2009	01/09/2008	Special Service	2	2	255	
5186091	08/01/2009 14:34:00	2009	01/09/2008	Special Service	2	2	255	
5363091	08/01/2009 19:23:00	2009	01/09/2008	Special Service	2	2	255	
5682091	09/01/2009 11:01:00	2009	01/09/2008	Special Service	2	2	255	
5688091	09/01/2009 11:18:00	2009	01/09/2008	Special Service	2	2	255	
5724091	09/01/2009 12:40:00	2009	01/09/2008	Special Service	2	2	255	
5770091	09/01/2009 13:43:00	2009	01/09/2008	Special Service	2	2	255	
5789091	09/01/2009 14:30:00	2009	01/09/2008	Special Service	2	2	255	
5797091	09/01/2009 14:39:00	2009	01/09/2008	Special Service	2	2	255	
6259091	10/01/2009 9:35:00	2009	01/09/2008	Special Service	2	2	255	
6270091	10/01/2009 10:09:00	2009	01/09/2008	Special Service	2	2	255	

Figura 8. Pantalla de previsualización de los datos que permite realizar ediciones.

En nuestro caso parece que el formato fecha, FinYear (fin del servicio), no lo ha importado correctamente, ya que tiene el mismo valor en todos los campos. Como no es un campo que utilizaremos, lo eliminamos a modo ilustrativo de las transformaciones que se pueden realizar con los datos en este momento.

Para ello seleccionamos la opción **Transformar datos** y se abrirá el gestor de datos integrado en Power BI denominado Query Manager.

Seleccionamos la columna y en el menú contextual la opción **Quitar**.

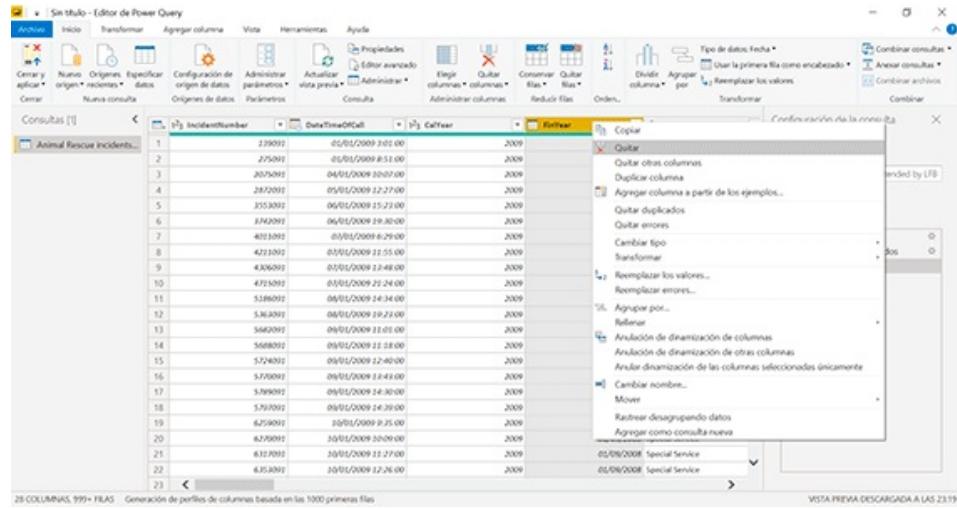


Figura 9. Query Manager. Eliminamos una columna.

También observamos que el campo **IncidentNumber** presenta un error de formato.

En este caso le aplicaremos el formato **Texto**.

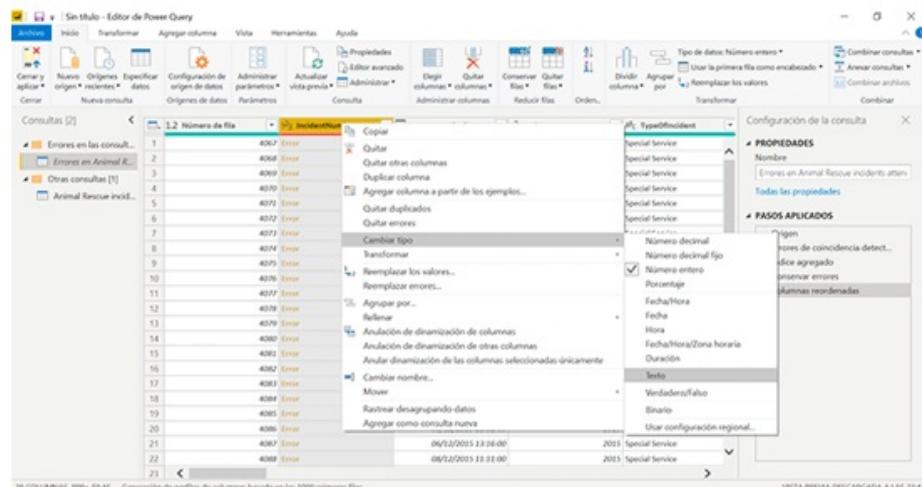


Figura 10. Query Manager. Cambiamos el formato de un campo.

El gestor de datos Query Manager nos ofrece multitud de opciones para transformar los datos, no solo la opción de eliminar columnas o cambiar formatos de los campos. También podemos reemplazar valores en un campo tipo fecha que nos deje solo el año, etc.

Una vez realizadas todas las transformaciones necesarias sobre los datos, pinchamos en la opción **Cerrar y aplicar** con el objetivo de hacer efectivos dichos cambios.

Podremos observar que los campos de nuestro *dataset* ya figuran en la columna Campos, ubicada en el menú derecho de nuestra pantalla de trabajo o Vista informe.

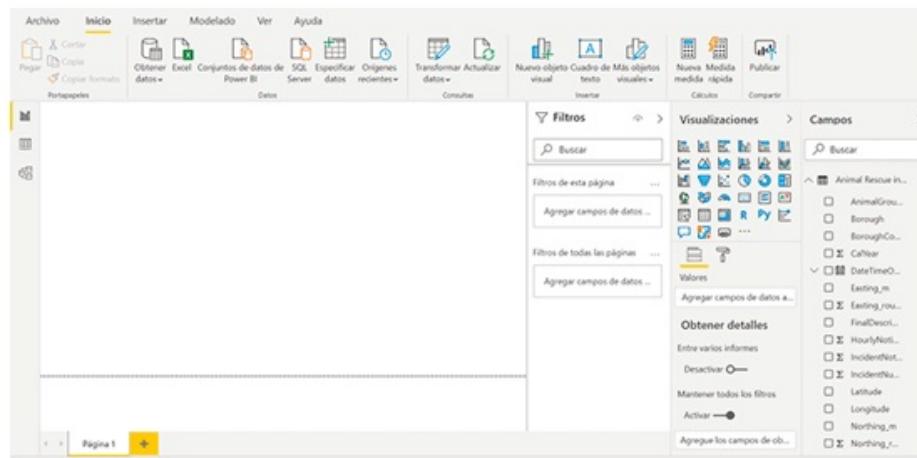


Figura 11. Dataset cargado en la vista Informe.

7.5. Visualización de datos

Con los datos ya cargados, el siguiente paso a seguir en el proyecto es la **generación de informes**.

Para crear una visualización tenemos que arrastrar el campo que queremos incluir en el informe a la **Vista Informe**. En ese momento también podemos seleccionar en el menú lateral derecho el tipo de visualización que más nos interese en función de los datos.

Si no escogemos ningún tipo de visualización, la que aplica por defecto es la tabla de datos.

En nuestro caso escogemos el tipo de animal rescatado, la ciudad, el código postal y la descripción de la intervención.

Figura 12. Visualización de datos en formato Tabla.

A medida que se van incluyendo los campos, en la columna de **Filtros** automáticamente aparecen los filtros y los valores posibles para cada uno de ellos. Por ejemplo, se puede observar que en el filtro de **Tipo de Animal**

(AnimalGroupParent) aparecen valores como *Bird* o *Cat*.

Ahora bien, ¿cómo podríamos saber cuánto gastan los bomberos londinenses (LFB) en salvar algunos tipos de mascotas? Podemos trabajar con hipótesis, pero al final hay que contrastarlas con datos. Para ello vamos a realizar una exploración previa inicial que nos ayude a identificar los campos relevantes y nos permita **segmentar los datos** eficazmente.

La primera hipótesis sería confirmar la tendencia en el número de servicios que realizan los bomberos por esta casuística. ¿Crecen, se mantiene o decrece? Utilizaremos la visualización **Line Chart** y arrastramos el sumatorio de CallYear bajo la etiqueta Ejes (para que aparezca en el eje horizontal), el campo PumCount lo llevamos bajo la etiqueta Values para el eje vertical.

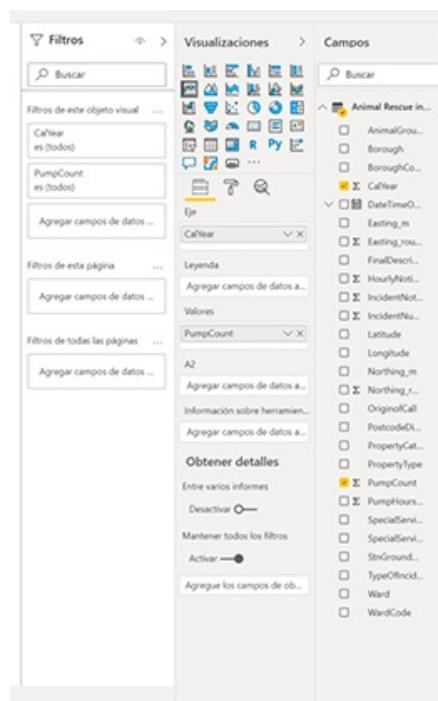


Figura 13. Selección de tipo de gráfica y campos para los ejes y valores.

Con estos simples pasos ya tenemos el primer informe con la evolución anual del número de servicios.

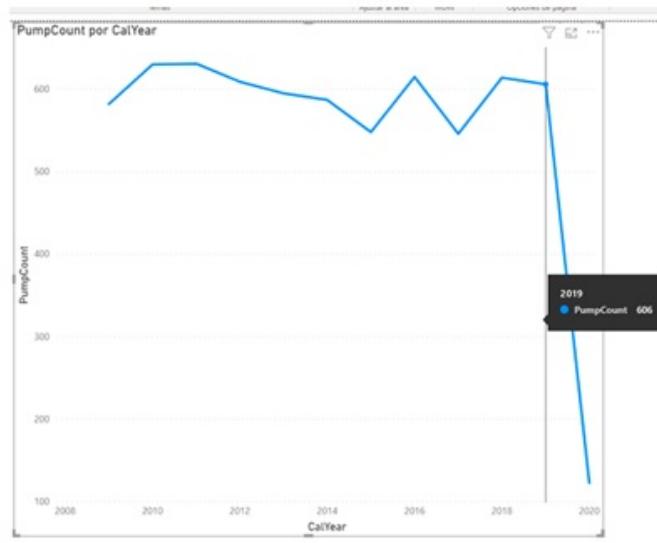


Figura 14. Evolución de las intervenciones anuales de rescate de mascotas.

Dado que para el año 2020 no tenemos todos los datos para todos los meses, podemos aplicar un **filtro que elimine dicho año** del análisis.

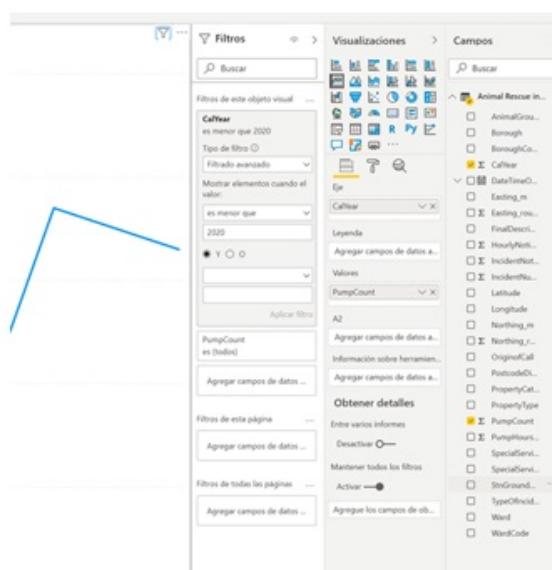


Figura 15. Evolución de los casos por año aplicando filtro.

También podríamos utilizar la visualización de tipo **Funnel**. Es tan sencillo como

seleccionar en el panel de control el nuevo tipo de gráfica. Para construir dicho *Funnel* por defecto nos ordena los años de mayor a menor número de intervenciones, pero esto nos mezclaría los años.

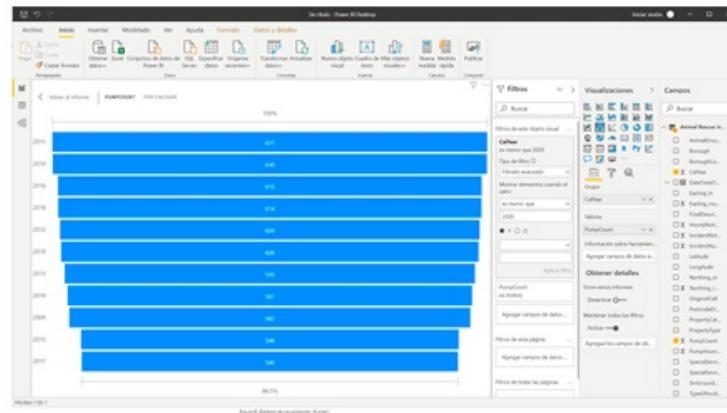


Figura 16. *Funnel* del número de intervenciones.

Todas las gráficas tienen la opción de ordenar los datos que se visualizan con base en diferentes criterios. En este caso nos interesa que las intervenciones aparezcan ordenadas en el tiempo.

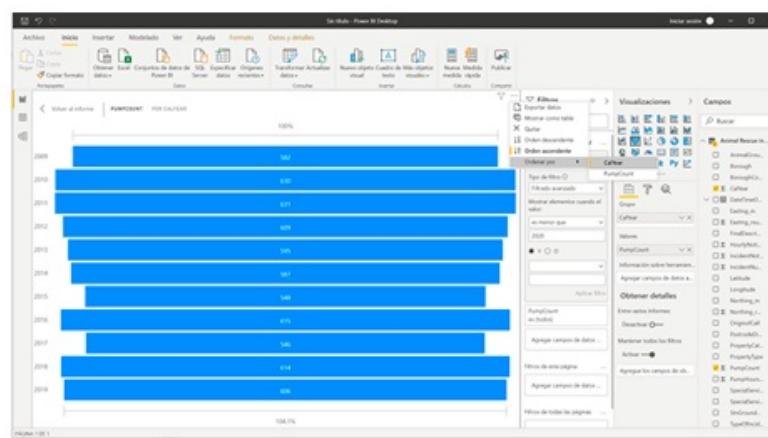


Figura 17. Orden de los datos de la gráfica.

Ahora sí podemos apreciar que entre los años 2009-2011 hubo un repunte de los casos, en el 2016 se produjo un pico y en 2018 volvieron a crecer los casos.

Veamos si estos repuntes anuales han tenido su reflejo en el coste del servicio.

Elaboramos una gráfica de área donde arrastramos el campo CalYear a la etiqueta Eje, y el campo IncidentNotionalCost a la etiqueta Valores.

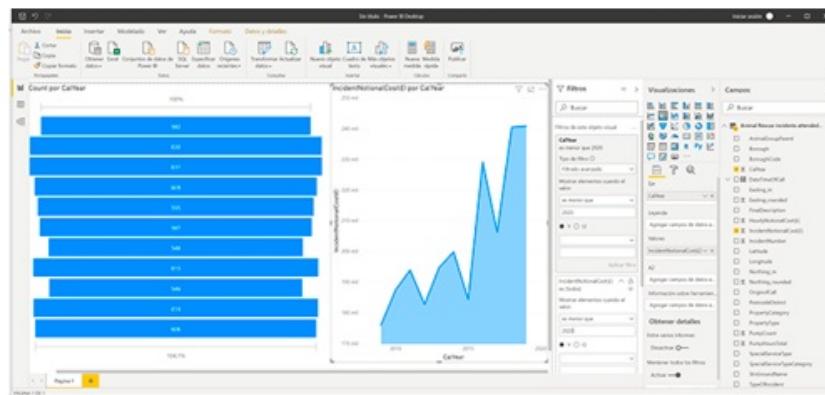


Figura 18. Número de incidentes y evolución coste del servicio.

Todo parece cuadrar salvo para el periodo 2012-2014, donde a pesar de disminuir los casos, el coste se incrementa año tras año. Sigamos buscando una explicación y analicemos cómo otros campos del *dataset* pueden influir en esta aparente contradicción.

Analicemos, por ejemplo, si el tipo de animal que ha requerido la intervención puede influir en el coste. Para ello identificaremos la distribución de los animales rescatados a lo largo del tiempo mediante una gráfica *pie chart*.

Arrastramos el campo AnimalGroupParent al campo Leyenda y CalYear al campo Valores. Podemos ver así, que el tipo de animal más frecuentemente rescatado (49,2 %) es el gato, seguido por los pájaros (19,09 %) y por los perros (16,7 %).

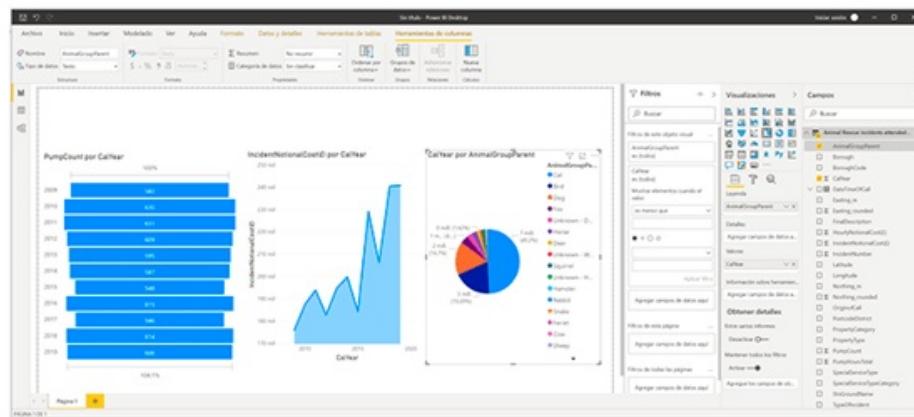


Figura 19. Distribución de los tipos de animales rescatados.

Pinchando en cada tipología de animales en el *pie chart*, podemos ver que las otras dos gráficas actualizan sus datos reflejando solo el número de incidentes y el coste del rescate asociado a dicha tipología de animal.

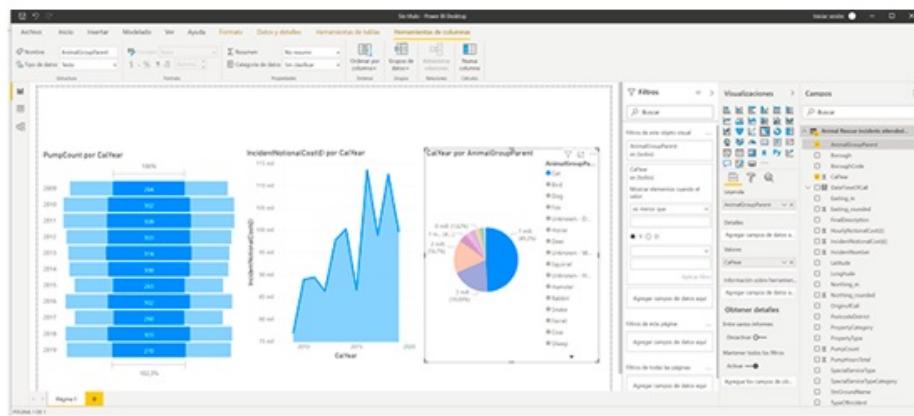


Figura 20. Datos actualizados de incidentes y coste según el tipo de animal seleccionado en el *pie chart*.

Esto nos puede ayudar a analizar las desviaciones. Y se puede ver que en el periodo que estamos analizando 2012-2014, la tendencia en coste del rescate de gatos aumentó, en el de los pájaros solo en 2013-2014 y en el de los perros solo en el 2012-2014.

7.6. Visualizaciones avanzadas

Power BI incorpora múltiples tipologías de gráficas, pero quizás la que siempre más interés despierta es la **distribución geográfica**.

En este ejemplo que estamos desarrollando, creamos una nueva solapa **Página 2** pinchando en la parte inferior del área de edición, y seleccionamos el tipo de gráfica **Mapa**. Llevamos a la etiqueta Ubicación el campo Borough (ciudad), a la etiqueta Leyenda, el campo AnimalGroupParent, y a la etiqueta Tamaño, al campo PumpCount.

Puede ser que aparezcan nombres de ciudades ubicadas también fuera de UK. Esto es porque dichas ciudades pueden existir en otras partes del mundo y tienen el mismo nombre. Basta con hacer clic en dichos lugares y eliminarlos de la visualización con la opción Excluir que aparece al pulsar el botón derecho del ratón.

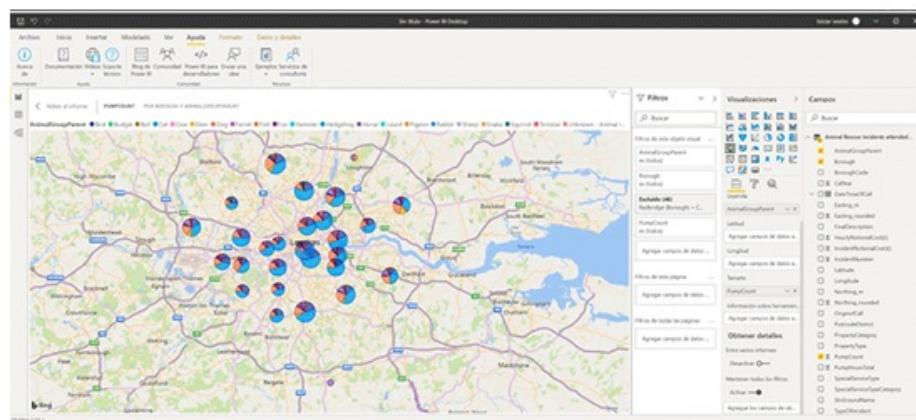


Figura 21. Mapa visual con la distribución de los rescates realizados.

Como podemos apreciar en la gráfica, la distribución de los rescates se despliega a través de todo el área metropolitana de Londres.

Power BI también ofrece una serie de recursos muy interesantes para avanzar en el

entendimiento de los datos. Por ejemplo, podemos segmentar los datos seleccionando un valor concreto.

Hemos visto que lo podemos hacer pinchando en cualquier de las representaciones del *pie chart*, pero también podemos incorporar lo que se denomina la visualización tipo **Segmentación de Datos**. Este tipo de visualización no es más que la activación de una serie de filtros con base en un campo determinado. Lo vemos en la siguiente gráfica donde hemos activado dicha visualización y hemos arrastrado a la etiqueta Campo el campo Borough.

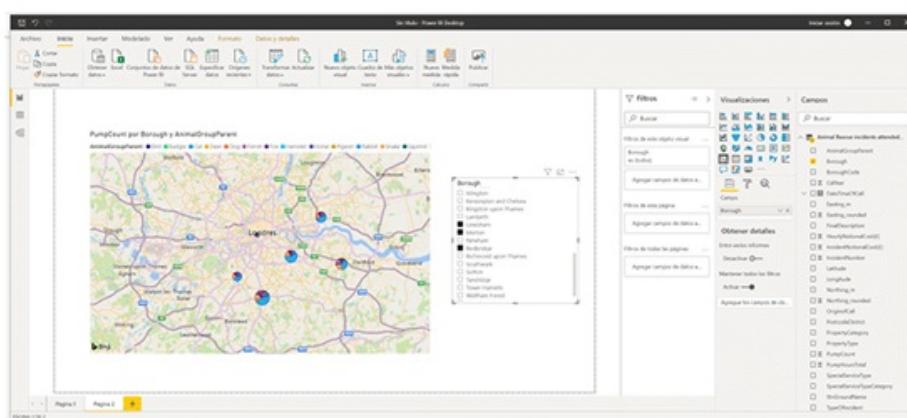
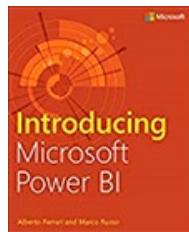


Figura 22. Visualización Segmentación de Datos.

En definitiva, a través de este caso de uso completo hemos desarrollado una aproximación a la capacidad de Power BI para el tratamiento de los datos y la visualización sencilla de los mismos.

Introducing Microsoft Power BI

Ferrari, A. y Russo, M. (2016). *Introducing Microsoft Power BI*. [S. l.]: Microsoft.



Este libro aborda de una forma sencilla cómo usar Power BI. Es una inspiración para mejorar la toma de decisiones basadas en datos, aprovechando las características analíticas y de colaboración de este entorno.

Applied Microsoft Power BI: Bring your data to life!

Lachev, T., Price, E. y Underwood, J. (2016). *Applied Microsoft Power BI: Bring your data to life!* (5^a ed.). [S. l.]: Prologika.



Este libro te ayudará a exprimir las capacidades de este entorno de análisis de datos y visualización, descubriendo cómo se integran y transforman los datos e implementando sofisticados modelos de análisis descriptivos y predictivos.

Making Cities Smarter: How Citizens' Collective Intelligence Can Guide Better Decision Making

Edggers, W. D., Green, M. y Guszcza, J. (2017). Making Cities Smarter: How Citizens' Collective Intelligence Can Guide Better Decision Making. *Deloitte Review*, 20, 139-153. Recuperado de <https://www2.deloitte.com/us/en/insights/deloitte-review/issue-20/people-for-smarter-cities-collective-intelligence-decision-making.html>

En este estudio nos sugiere cómo tener en cuenta la inteligencia colectiva a través del análisis de datos.

Microsoft Power BI Community

Community blog. Microsoft Power BI Community. Recuperado d e https://community.powerbi.com/t5/Community-Blog/ct-p/PBI_Comm_CommunityBlog

Blog y foro de la comunidad de Power BI en el sitio oficial de Microsoft.

Microsoft Power BI Community > **Community Blog**

1. Power BI es una herramienta que permite la visualización de datos:

 - A. Correcto. No incorpora capacidades para el tratamiento y transformación de los datos.
 - B. Incorrecto. Solo la versión Desktop permite la visualización de datos.
 - C. Incorrecto. Solo la versión SaaS permite la visualización de datos.
 - D. Correcto. Adicionalmente incorpora capacidades avanzadas para el tratamiento y transformación de los datos.
2. Power BI permite el intercambio rápido entre la visualización de datos y la visualización de gráficas:

 - A. Correcto. Desde el menú lateral izquierdo es posible intercambiar dichas vistas.
 - B. Correcto. Pulsando F5.
 - C. Incorrecto. Una vez cargados los datos, ya solo nos movemos en su visualización.
 - D. Incorrecto. Existe la opción de carga de datos y visualización posterior en modo gráfica. La visualización de los datos no es necesaria en el 80 % de las ocasiones.
3. ¿Dónde se ubican las opciones de Filtros, Visualizaciones y Campos?

 - A. En el menú lateral derecho.
 - B. No existe el menú de Filtros.
 - C. No existe el menú de Visualizaciones.
 - D. No existe el menú de Campos.

4. ¿Qué fases ordenadas incorpora un proyecto de análisis y visualización de datos?

 - A. Preparación – Obtención - Modelado – Visualización – Reporte de datos.
 - B. Obtención – Modelado – Preparación – Modelado – Visualización – Reporte de datos.
 - C. Obtención – Preparación – Modelado – Visualización – Reporte de datos.
 - D. Obtención – Visualización de datos.
5. El gestor de datos integrado de Power BI:

 - A. Permite limpiar datos.
 - B. Permite tratar los datos.
 - C. Se denomina Query Manager.
 - D. Todas las respuestas son correctas.
6. Los filtros dinámicos de Power BI nos permiten seleccionar el rango deseado de datos, tanto desde el apartado Filtros como incorporando una visualización de Segmentación de Datos:

 - A. Correcto.
 - B. Incorrecto. No existe la visualización Segmentación de Datos.
 - C. Incorrecto. La visualización Segmentación de Datos no tiene esa finalidad.
 - D. Incorrecto. El apartado Filtros sirve para escoger el tipo de gráfica deseada.

- 7.** En el momento de cargar los datos es posible editarlos antes de proceder a su visualización:
- A. Correcto. En este momento podemos utilizar la opción editar para hacer los cambios de formato que sean necesarios.
 - B. Correcto. Sin embargo, si modificamos los datos debemos guardarlos y volver a cargarlos para que recoja todas las modificaciones.
 - C. Incorrecto. Power BI no incorpora dicha funcionalidad.
 - D. Incorrecto. Power BI permite la edición de datos pero no antes de analizarlos visualmente.
- 8.** Las visualizaciones tipo Mapa requiere la presencia de coordenadas en los datos:
- A. Correcto. No existe otra manera de hacerlo.
 - B. No es necesario, porque el servicio de mapas de Power BI también es capaz de interpretar localidades en formato texto.
 - C. No es necesario, porque basta con indicárselo en los Filtros.
 - D. Correcto. Las coordenadas GIS deben estar presentes en el *dataset*.
- 9.** La gráfica tipo Funnel solo se aplica si los datos de los diferentes niveles tienen un orden decreciente formando un embudo:
- A. Incorrecto. El orden de los datos no es relevante y se pueden aplicar diferentes criterios para ordenarlos.
 - B. Incorrecto. Tienen que estar ordenados de forma creciente para forma el embudo.
 - C. Correcto. Debes estar ordenados decrecientemente.
 - D. Correcto. El embudo solo muestra datos que representen variaciones entre ellos.

10. Los campos de los ejes y de los valores son definidos automáticamente por Power BI:

- A. Correcto. Así nos facilita la experiencia como usuarios.
- B. Incorrecto. Son definidos por el usuario.
- C. Correcto y no se pueden modificar.
- D. Ninguna de las anteriores.

Herramientas de Visualización

Tema 8. Qlik Sense

Índice

[Esquema](#)

[Ideas clave](#)

[8.1. Introducción y objetivos](#)

[8.2. Instalación de Qlik Sense](#)

[8.3. Apps](#)

[8.4. Exploración de los datos, filtros y tablas pivotantes](#)

[8.5. Etiquetas y colores](#)

[8.6. Mapas geográficos](#)

[8.7. Historias](#)

[8.8. Otros gráficos avanzados](#)

[A fondo](#)

[Learning Qlik Sense](#)

[Mastering Qlik Sense](#)

[Onboarding Qlik Sense](#)

[Qlik vídeos](#)

[Biblioteca de recursos](#)

[Test](#)

Esquema



8.1. Introducción y objetivos

Qlik Sense es una plataforma completa orientada al descubrimiento de la información que reside en los datos. Eso significa que no solo estamos ante una herramienta de visualización, sino que incluye capacidades propias de entornos de *Business Intelligence*. Es decir, Qlik nos permite **recopilar, procesar y extraer valor de los datos**.

Accede a la página web oficial de Qlik a través del siguiente enlace: <https://www.qlik.com/es-es/products/qlik-sense>

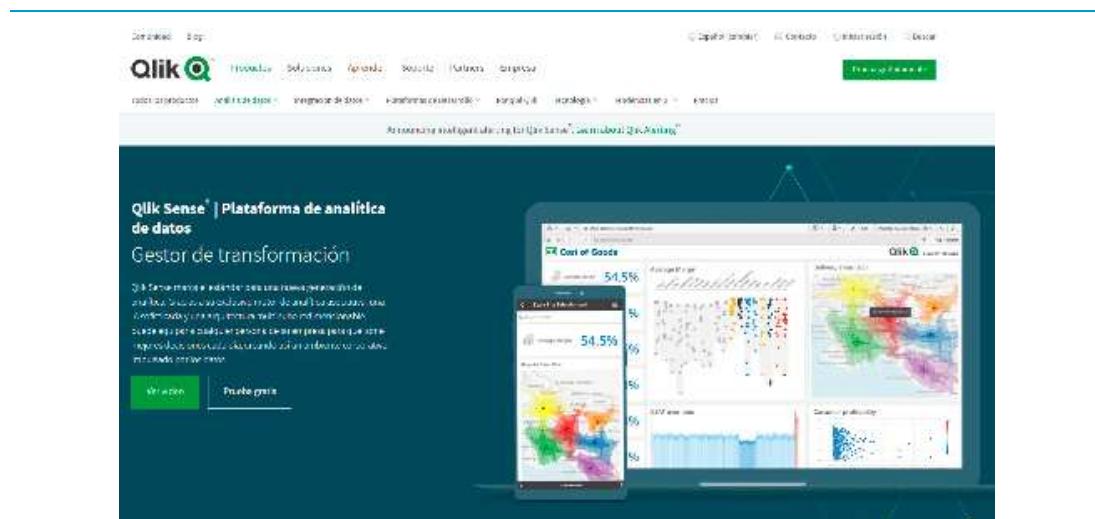


Figura 1. Qlik Sense. Plataforma de analítica de datos.

Empezaremos desde cero con el objetivo de crear visualizaciones de forma sencilla e intuitiva. Comenzando por gráficas simples, ajustando tamaños, filtros, etc. Y poco a poco, iremos incorporando más elementos.

Qlik Sense es un entorno visual que nos facilita la creación de gráficos mediante el modo **«arrastrar y soltar»**.

8.2. Instalación de Qlik Sense

Qlik Sense tiene diferentes versiones: *business* y *enterprise*:

- ▶ La versión ***business*** es la versión orientada a pequeñas y medianas empresas y difieren en sus capacidades como, por ejemplo, posibilidades de integración con APIs, número de espacios o áreas para almacenamiento y desarrollo de apps compartidos, límites de espacio en la app y número de recargas de datos.
- ▶ En la versión *business* no viene incluidas algunas funcionalidades avanzadas que sí están disponibles en la versión ***enterprise*** como Qlik Nprinting (informes para distribuir), Qlik Associative Big Data Index (conexión a *datasets* en la nube) o Qlik GeoAnalytics (funciones avanzadas para geoanalítica).

A lo largo de la asignatura emplearemos la versión *business*, que permite la creación de apps y visualizaciones, tanto en su versión *desktop* como *cloud*, de la misma forma que la versión *enterprise*. Dispone de versiones de prueba para poder evaluar y probar todos los ejemplos de la asignatura.

Accede a la versión de prueba a través del siguiente enlace: <https://www.qlik.com/us/>

En todo caso, las capacidades de versiones *desktop* y *cloud*, aunque difieran ligeramente en su interfaz, son excepcionales y la forma de crear visualizaciones es prácticamente la misma.

8.3. Apps

Para crear una nueva app basta con ir a la página de inicio, descartar los paneles informativos que aparecen la primera vez y podremos observar un mensaje inicial diciéndonos que no tenemos aún creado ningún contenido.

Pulsamos en el botón **Crear nueva app**. Tras darle un nombre a la app, en este caso «Mi primera App», tendremos ya nuestra primera aplicación:

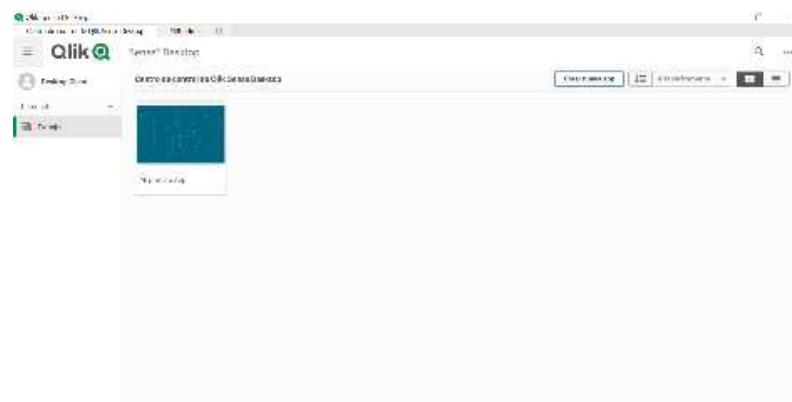


Figura 2. Entorno de apps en Qlik Sense.

Hacemos clic en la app y abrimos la aplicación donde podremos empezar a añadir datos de archivos locales y de otras fuentes de datos a nuestro *dashboard*.

Añadir datos a una app

Lo primero que podemos hacer con una app es indicar qué datos vamos a utilizar para crear la visualización. Existen dos opciones básicas:

- ▶ Utilizar un **fichero de datos** que tengamos descargado en nuestro ordenador.
 - ▶ Utilizar **fuentes de datos externas** mediante la utilización de los múltiples conectores que nos proporciona Qlik.

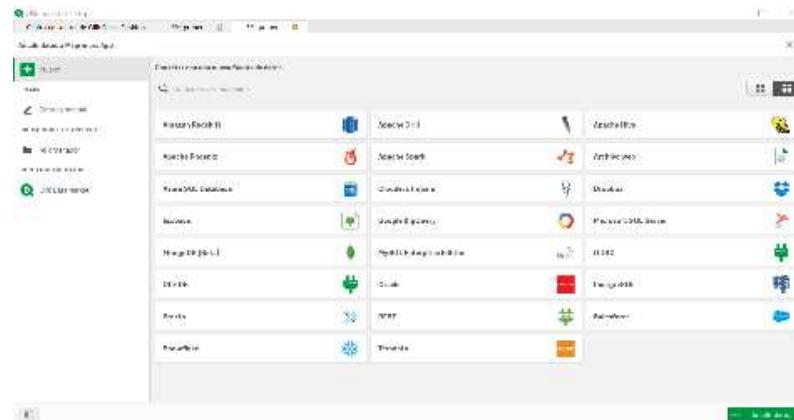


Figura 3. Conectores de fuentes de datos.

En nuestra primera app vamos a utilizar datos de ejemplo que encontraremos en la opción Qlik Datamarket, presente en la versión *desktop*, que no es más que un repositorio de datos reales, algunos de ellos de pago. En concreto emplearemos un *dataset* con datos de economía que contienen la distribución del PIB por países a nivel mundial.

En Internet existen múltiples sitios donde poder encontrar diferentes *datasets*. Una fuente de libre acceso muy interesante es el Banco Mundial.

Accede a la página oficial del Banco Mundial a través del siguiente enlace:

<http://datos.bancomundial.org>

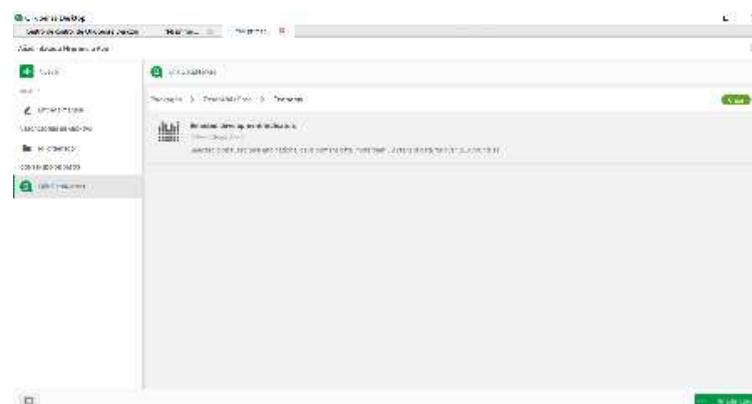


Figura 4. Acceso a *datasets* predefinidos en Qlik Sense Desktop.

Una vez seleccionados los datos, aparecerá una pantalla donde se nos indica el modelo de datos del fichero escogido. En este ejemplo, veremos dos tablas unidas por un campo, y también podremos observar la estructura de las dos tablas.

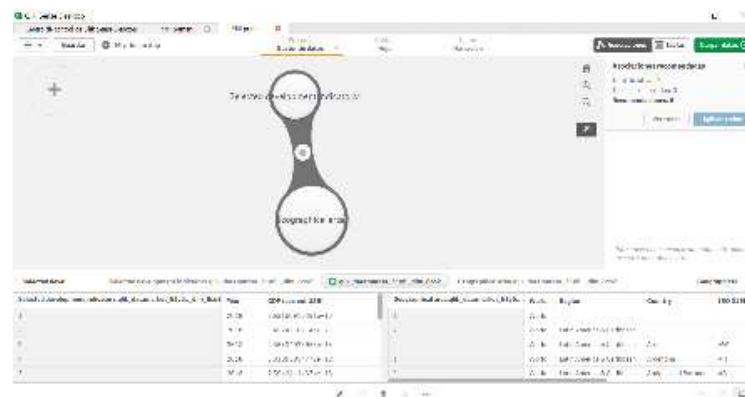


Figura 5. Relaciones entre tablas.

A continuación, pulsamos en Cargar datos y ya podemos ir a Editar la hoja. En nuestra hoja, podemos escoger el tipo de gráfico y tenemos que elegir una dimensión y una medida.

Dimensiones y medidas

A la hora de crear un gráfico Qlik Sense nos pide la dimensión y la medida. Utilizamos una combinación de ambas para llenar de datos las visualizaciones.

- ▶ Dimensión: es la **característica del dato**. Nos ayuda a determinar cómo agruparemos los datos de la medida. Por ejemplo: fecha, ciudad, edad, ventas, productos...
- ▶ Media: es un **cálculo**. Lo utilizamos para decidir qué parte de los datos mostrar en la visualización. Por ejemplo: suma del total de ventas o media del número de productos.

En la siguiente imagen, el tipo de producto es la dimensión y la suma del coste es la medida.

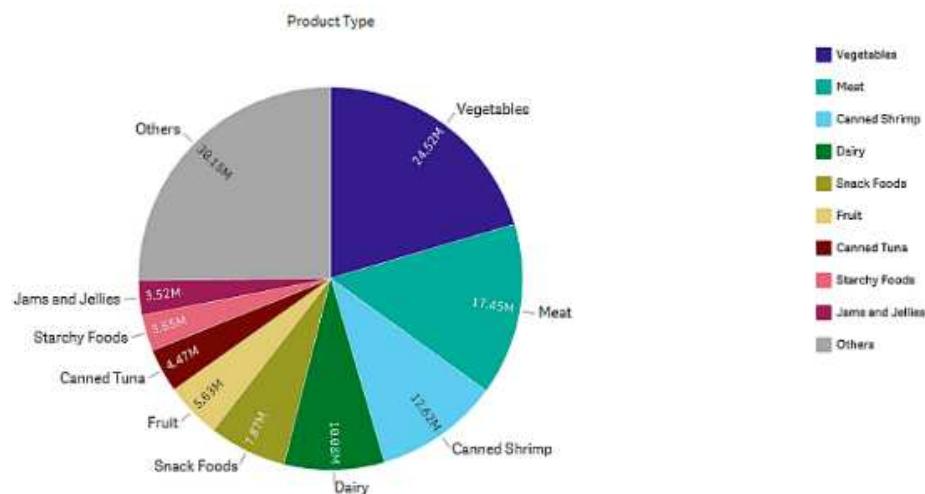


Figura 6. Ejemplo de dimensión y medida.

En esta otra gráfica, la dimensión es doble: año y región. La medida es la suma de las ventas.

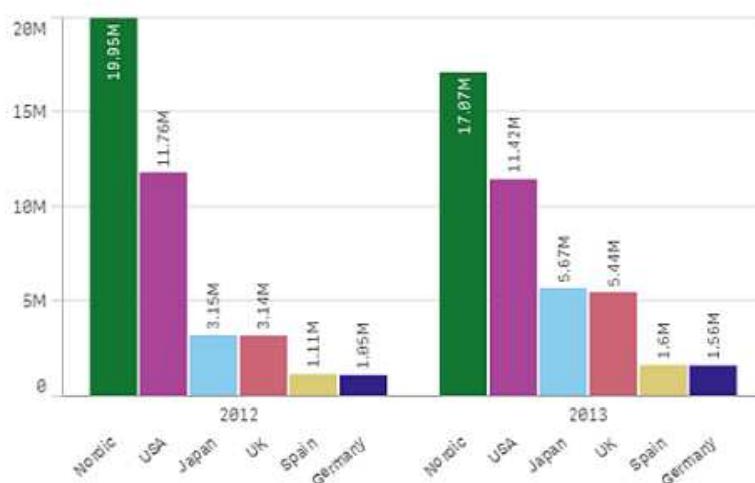


Figura 7. Doble dimensión: año y región.

Entendido el significado de estos dos elementos, continuamos con nuestro ejemplo

anterior. Seleccionamos como tipo de gráfico un diagrama de barras, como dimensión el campo Country y como medida el campo la suma del GPD (PIB en inglés).

Ya tenemos un gráfico sencillo, pero totalmente funcional con unos pocos clics.

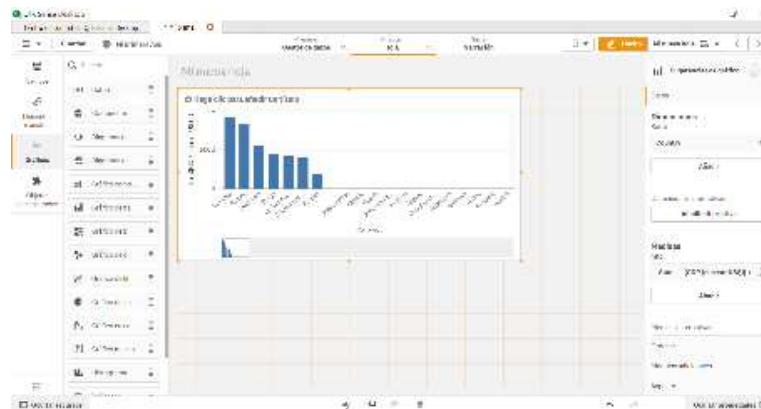


Figura 8. Gráfico de barra.

También podemos incluir **filtros** al gráfico, a través de la opción del menú lateral Panel de filtrado, arrastrando la opción del menú lateral a la zona de edición. Podemos escoger, por ejemplo, una dimensión por país.

Pinchando en la opción Hecho, marcada con un lápiz naranja en la parte superior del menú, visualizamos la gráfica y podremos emplear dichos filtros.

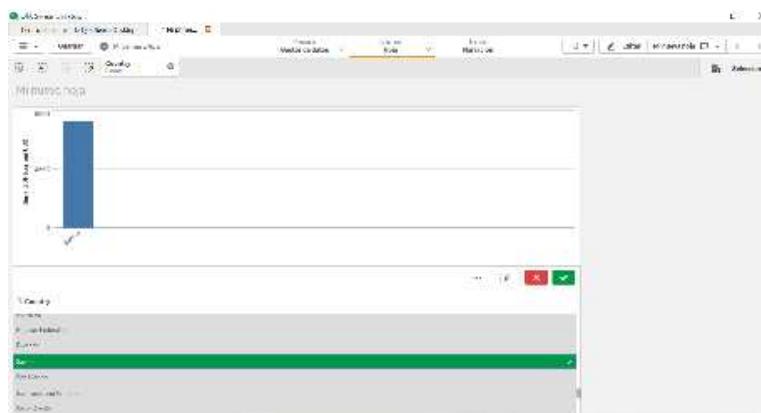


Figura 9. Ejemplo de filtros.

Es una primera aproximación para que veas la dinámica de cómo funciona Qlik Sense y cómo crear una app de forma muy rápida.

Gestión de la app

Vamos a ver la gestión que podemos hacer de la app a través de las principales opciones del menú superior.



Figura 10. Menú de operaciones de la app.

Vista general de la app

Pinchando en la primera opción del menú superior



podemos ir a la Vista general de la app, donde estarán las hojas que tenemos. En este caso, hemos creado una hoja y nos permite renombrarla, crear una nueva, etc.

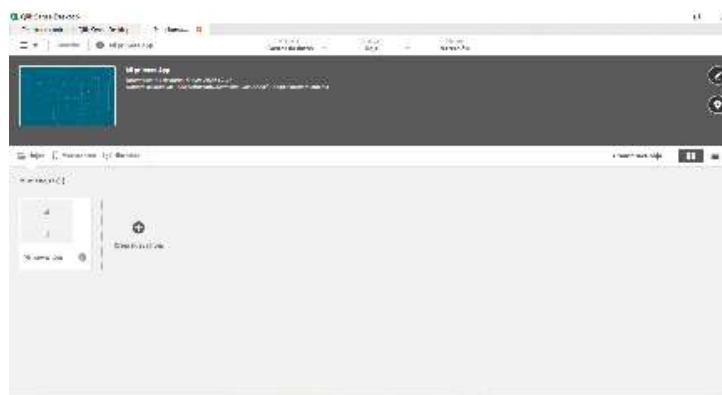


Figura 11. Vista general de la app.

Gestor de datos

Esta opción nos permite ver la estructura de nuestras tablas y las relaciones

existentes entre ellas a través de los denominados **campos clave**. Son los campos comunes en las tablas que nos permiten asociar unas tablas con otras y que ya hemos visto en el momento de cargar los datos a nuestra primera app.

Además de las posibles asociaciones entre tablas, también podemos trabajar con sus datos, ordenándolos, dividiendo las tablas, reemplazando valores, etc.

Editor carga de datos

Esta opción nos permite trabajar con el código que se utiliza para cargar los datos. Al crear nuestra primera app no hemos tenido que crear ni utilizar código, ya lo ha hecho Qlik Sense por nosotros con su gestor de datos. Detrás de una carga de datos, Qlik siempre elabora el código necesario para el tratamiento de estos.

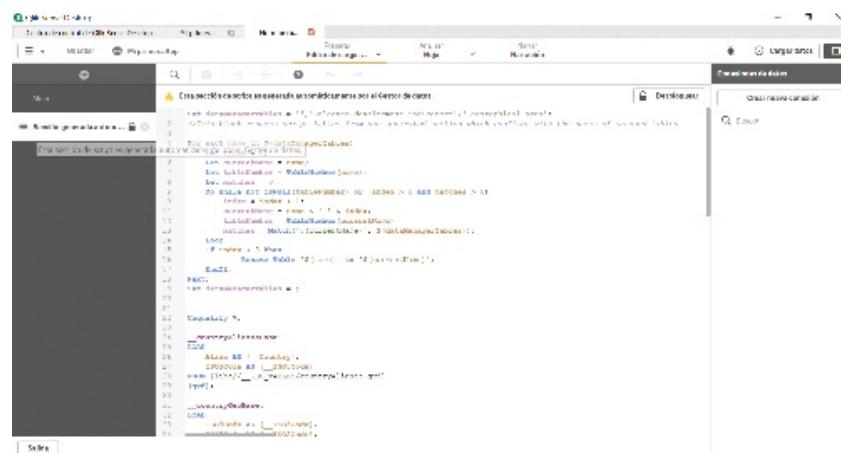


Figura 12. Script creado automáticamente con código para el tratamiento de los datos.

Esta sección por defecto está bloqueada, pero se pueden modificar los *scripts* pulsando en el botón Desbloquear.

Visor del modelo de datos

Es otra forma de ver los datos en un modelo. En nuestro ejemplo es muy simple porque solo tenemos dos tablas, pero si el *dataset* fuera más complejo o tuviéramos varias fuentes de datos cargadas, observaríamos las relaciones entre todas las

tablas y sus dependencias.

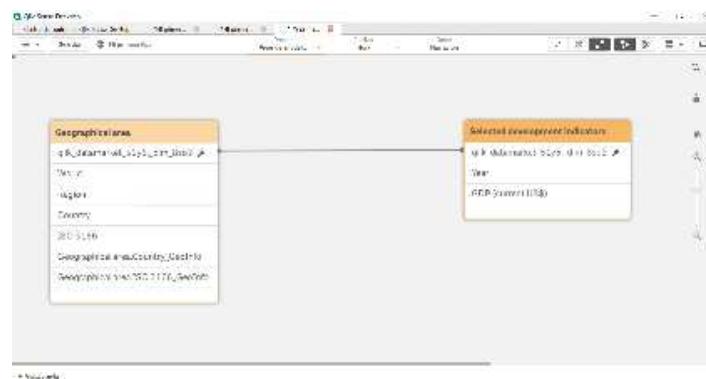


Figura 13. Modelo de datos.

8.4. Exploración de los datos, filtros y tablas pivotantes

Hemos visto los primeros pasos a dar con una app: cómo se crea, cómo se ven los datos y, en definitiva, una primera aproximación al mundo de Qlik Sense.

Avancemos explorando más opciones que nos ofrece esta herramienta. Para ello nos crearemos una segunda app y volvemos a añadir datos de Qlik Data Marketplace. En esta ocasión emplearemos datos demográficos y, en concreto, de población por país.

Una vez añadidos, aparecerá el gestor de datos que ya conocemos mostrando cuatro tablas (edad, país, sexo y población) y sus asociaciones.



Figura 14. Relaciones entre 4 tablas.

También podemos explorar las tablas y sus relaciones en un formato más tradicional.

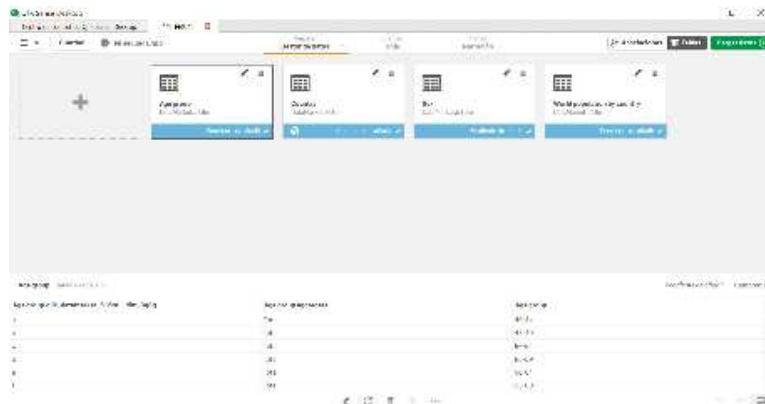


Figura 15. Detalle de las tablas.

Cargamos los datos y volvemos a editar la hoja. Creamos un gráfico de barras, con la dimensión país y con la medida, personas. Con ello creamos el gráfico que vemos en la figura 16.



Figura 16. Diagrama de barras con dimensión país.

Podemos cambiar el formato de los ejes, ya que no queda claro qué es 6G, 4G, 2G... en el eje de las Y. Seleccionamos en el lateral derecho el formato numérico y ya visualizamos medidas de forma más entendible.

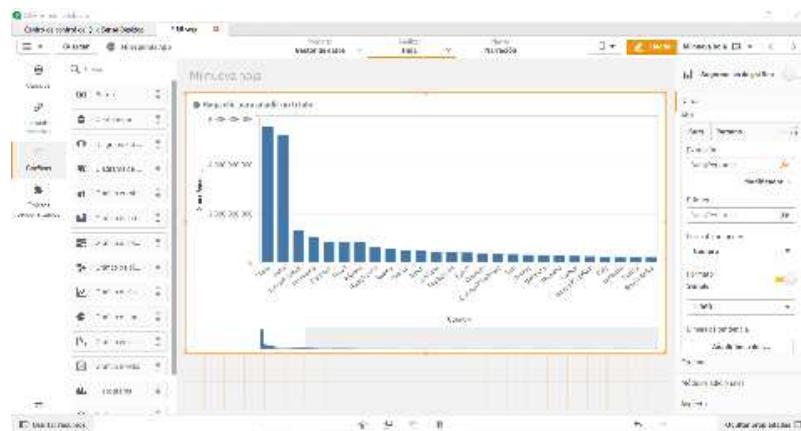


Figura 17. Modificación del formato de los ejes de una gráfica.

Con este cambio ya vemos que la unidad del eje Y eran miles de millones. Según la gráfica, China tendría 6000 millones de habitantes. Esto evidentemente no es posible, porque el mundo tiene algo menos de 8000 millones de habitantes. Por lo tanto, el próximo paso que vamos a abordar es la **exploración de los datos**.

Y para ello nuevamente acudimos a los filtros, incluyendo en esta ocasión las dimensiones de pais, sexo y edad. Pulsamos en el botón Hecho de la parte superior para probar dichos filtros.

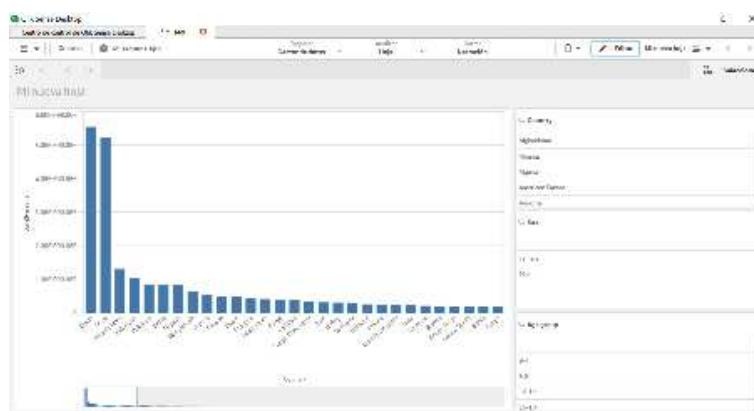


Figura 18. Filtros de una gráfica.

Examinando los filtros, vemos que en el campo sexo tenemos un campo vacío, al

igual que en grupos de edad. Si los seleccionamos, la cifra global de habitantes de China se sitúa en algo menos de 1400 millones de habitantes, lo que se ajusta a la realidad y tiene más sentido.

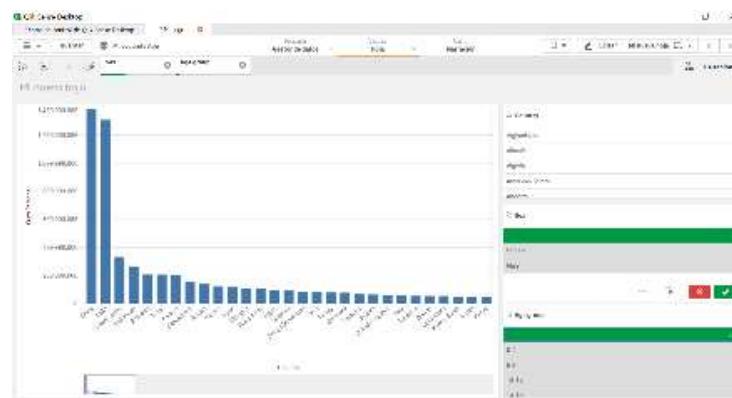


Figura 19. Selección de campos vacíos.

Volvemos a editar la hoja pinchando en la opción Editar y para verificar estos filtros, vamos a utilizar las tablas pivotantes.

Las **tablas pivotantes** se encuentran en el menú lateral izquierdo. Arrastramos esta opción al área de edición y seleccionamos como dimensión el campo grupos de edad y como medida, la suma de personas.

En la siguiente imagen se puede apreciar cómo existe un campo vacío en grupos de edades, que tiene más de 15 000 millones de habitantes. Efectivamente, esta exploración de los datos nos ha identificado el problema.

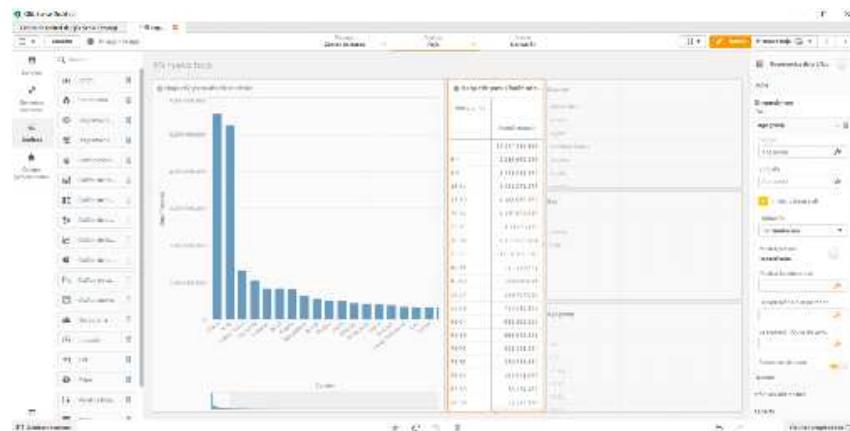


Figura 20. Tabla pivotante.

Podríamos hacer lo mismo con una segunda tabla pivotante para explorar el valor del campo sexo, ya que como hemos visto anteriormente en los filtros, también existen valores vacíos para este campo.

Por último, podemos almacenar el resultado de aplicar estos filtros utilizando un recurso muy interesante, que es el de **marcadores**. Con los filtros activados y habiendo seleccionado los campos vacíos en los filtros de grupos de edad y sexo, pinchamos en la parte superior en la opción de marcadores y le damos nombre a este nuevo marcador.

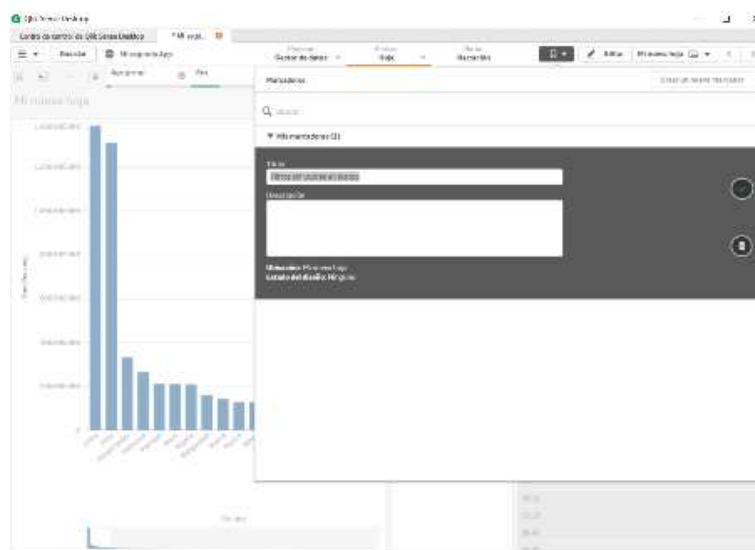


Figura 21. Creación de un marcador.

A partir de este momento, cada vez que vayamos a la opción de marcadores, podremos seleccionar el marcador creado y aplicará automáticamente los filtros incluidos en él.

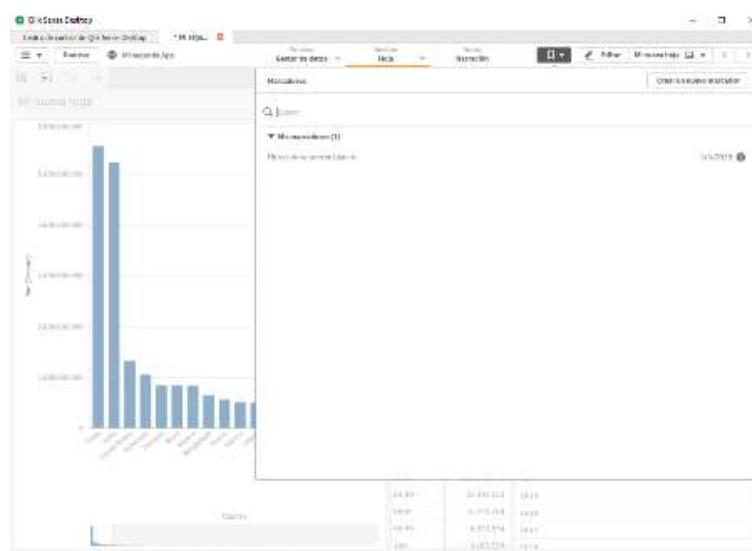


Figura 22. Selección de marcadores.

8.5. Etiquetas y colores

En esta sección vamos a centrarnos en mejorar la visualización de nuestra gráfica.

Para ello Qlik Sense pone a nuestra disposición múltiples recursos como las etiquetas, códigos de colores o la orientación de la gráfica.

Las **etiquetas** son siempre un recurso sencillo que mejora sustancialmente el entendimiento de los datos. Nuestra gráfica asigna a los ejes, por defecto, el nombre del campo que están representando y no necesariamente son ilustrativos.

Para modificarlos utilizaremos la opción Etiqueta, ubicada en el menú lateral derecho en el bloque de Datos. De esta forma, donde pone «Sum(Persons)» pondremos simplemente «Población», y donde pone «Country» pondremos «País».

En el bloque Aspecto, disponemos de la opción **Opciones de estilo**, que nos permite cambiar la **orientación** del gráfico. En este caso vamos a optar por un gráfico de barras horizontal.

También podemos cambiar los **colores** y la **leyenda del gráfico**. Por defecto los colores vienen en automático, si deshabilitamos esta opción, se despliega un abanico de opciones. Al escoger la opción Por dimensión, veremos que cada país toma un color diferente. También existe la opción Por medida, que asigna un gradiente secuencial a los valores de los países.

El resultado de estos cambios es el siguiente:

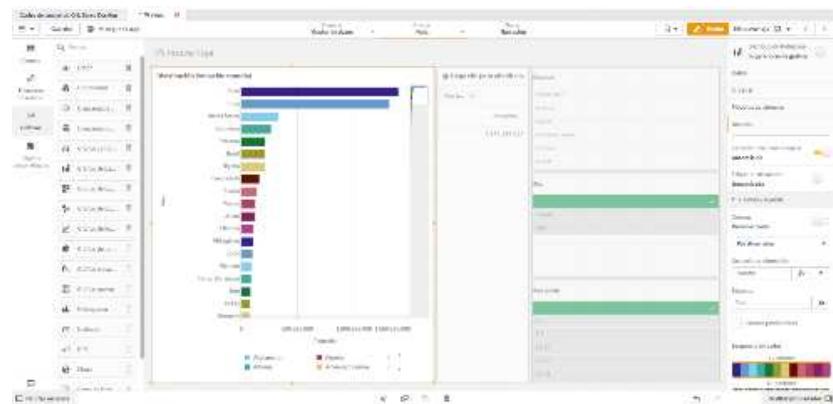


Figura 23. Opciones de etiquetas y colores de una visualización.

8.6. Mapas geográficos

Los mapas geográficos ayudan a representar la información de una forma visual y siempre son muy agradecidos por su sencillez y utilidad.

Evidentemente se necesita información geográfica que podamos representar, datos que estén mapeados con su longitud y latitud. Este es nuestro caso con el ejemplo que estamos desarrollando.

Para crear este tipo de representación arrastramos a la zona de edición la opción **Mapa** del menú lateral izquierdo.

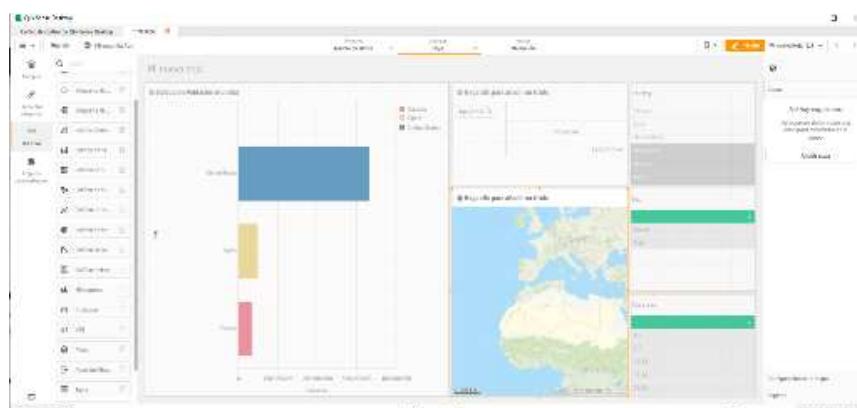


Figura 24. Mapa geográfico.

Este mapa aún no nos muestra ningún dato, porque no tiene datos asociados. Para hacerlo, basta con ir a la opción **Capas** del menú lateral derecho y seleccionar la opción **Añadir capa**. Existen diferentes tipos, nosotros seleccionamos Capa de área porque no estamos intentando representar puntos concretos en el mapa, sino que queremos visualizar zonas enteras que son los países.

A continuación, añadimos la **dimensión ISO 3166**, que es un dato que existe en nuestro *dataset*. ISO 3166 es un estándar internacional para los códigos de país de la Organización Internacional de Normalización. Con esta simple selección, si

pinchamos en el mapa en el modo visualización, ya podremos ver los datos asociados al país en el que hemos hecho clic.

Podemos cambiar el color de los países, asignando un color diferente a cada uno, seleccionando nuestra dimensión ISO 3166 y escogiendo la opción **Colores**. Nuevamente podremos asignar un color diferente a cada país o, lo que es más lógico en este caso, asignarles un color en función del número de habitantes. Para ello escogemos la opción Por medida y, al pinchar en el ícono de la fórmula,



se abrirá una nueva pantalla donde seleccionaremos el campo Personas y, en la Función de agregación, la opción Sum.

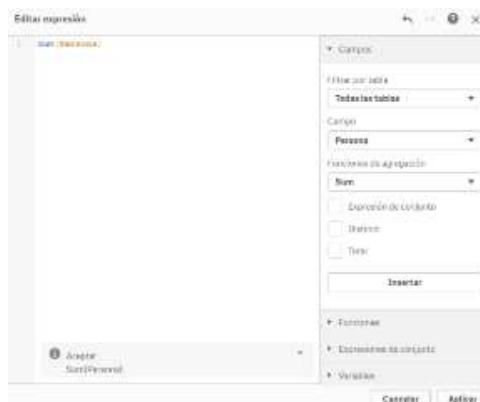


Figura 25. Función de agregación.

Ahora ya podemos ver el mapa con el código de colores por país aplicado.



Figura 26. Mapa geográfico personalizado con gradiente de colores.

Si quisieramos ver el nº de habitantes con más de 100 años, bastaría con seleccionar en el filtro el grupo de edad correspondiente, y veríamos que el dato se actualiza en todas las ventanas, incluido el mapa geográfico. Respondiendo a la pregunta, el país con un mayor número de personas con edad superior a los 100 años es Japón.

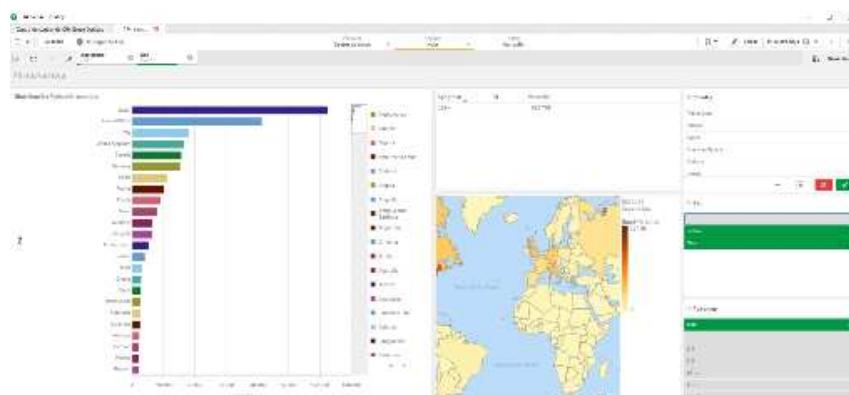


Figura 27. Mapa geográfico con filtros seleccionados.

8.7. Historias

Una opción importante de cualquier visualización es poder contar una historia con tus datos. Una historia es el equivalente a un *dashboard*. Para ello basta con seleccionar la opción **Historias** en el detalle de cada app, y pinchar en Crear nueva historia.

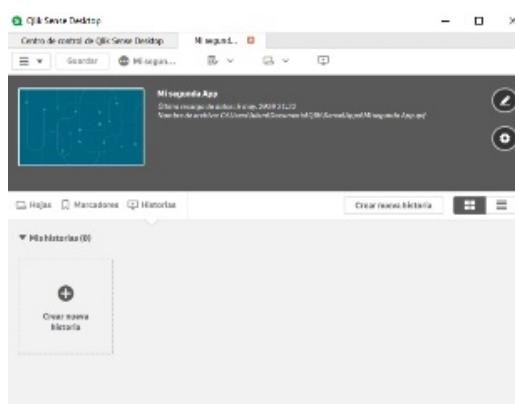


Figura 28. Historias de una app.

Lo que tenemos es un lienzo en blanco donde podemos ubicar títulos, textos explicativos, imágenes y evidentemente, gráficos. Cada lienzo es el equivalente a una diapositiva en PowerPoint. Podemos duplicarlas, borrarlas, etc.

Para insertar cualquier gráfica de las que tenemos, tendremos que seleccionar la imagen de la Librería de capturas, ubicada en el menú lateral derecho. Y para incluir una imagen en la librería, tenemos que ir a la gráfica que queramos seleccionar, hacer clic en el botón derecho del ratón y escoger el ícono de la cámara de fotos.

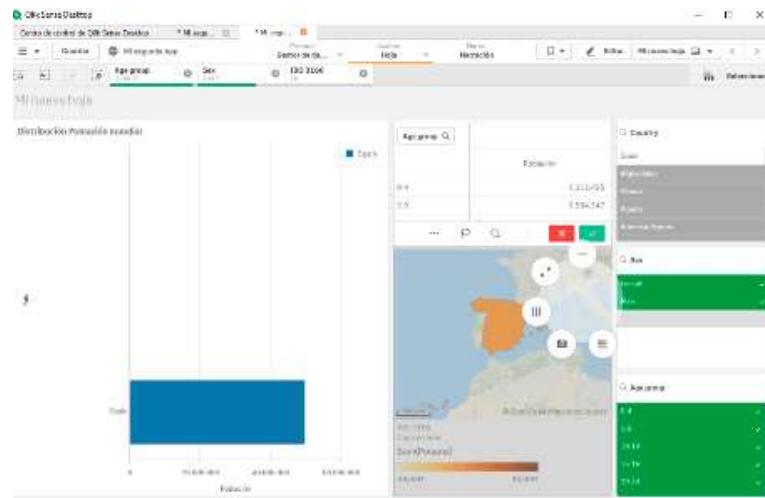


Figura 29. Opciones de una gráfica, incluida la captura de la imagen.

Con estas capturas, volvemos a la historia que estábamos creando y ya podemos incluir las imágenes desde la librería, añadir texto, formas, etc.



Figura 30. Creación de una historia.

Para ver la historia a modo de presentación hacemos clic en el botón verde ubicado en la parte superior izquierda, y ya tenemos nuestra propia versión de los datos en Qlik Sense.

8.8. Otros gráficos avanzados

Qlik Sense se caracteriza por una gran variedad de gráficos. En los ejemplos anteriores nos hemos basado en el gráfico de barras habitual, pero existen muchos más que veremos con un ejemplo de visualización.

Realmente lo relevante no es escoger el tipo de gráfico, sino dominar los conceptos para crear una app, analizar los datos, visualizarlos con filtros, crear historias, etc.

En todo caso, en el siguiente enlace se muestran diferentes tipos de gráficos en Qlik Sense para que veáis su potencial y algunas de las opciones de tipos de gráficos. Se ha escogido un análisis realizado sobre la maratón de Barcelona.

Accede a los datos sobre la maratón de Barcelona a través del siguiente enlace: <https://webappsqlik.com/barnamarato/index.html>

En esta primera visualización vemos cómo se mezclan los indicadores (KPI) con gráficas de barras y de línea.



Figura 31. Visualización combinada de indicadores y gráficas.

A continuación, se incluyen **tablas de datos** (perfectas para definir un *ranking*), junto con **histogramas** que nos ayudan a ver la distribución de las personas que han llegado a meta respecto al tiempo que han tardado.



Figura 32. Tablas de datos e histogramas.

Seguidamente, se vuelven a utilizar histogramas, gráficos de barras y un gráfico de tarta para representar los datos demográficos de los participantes.

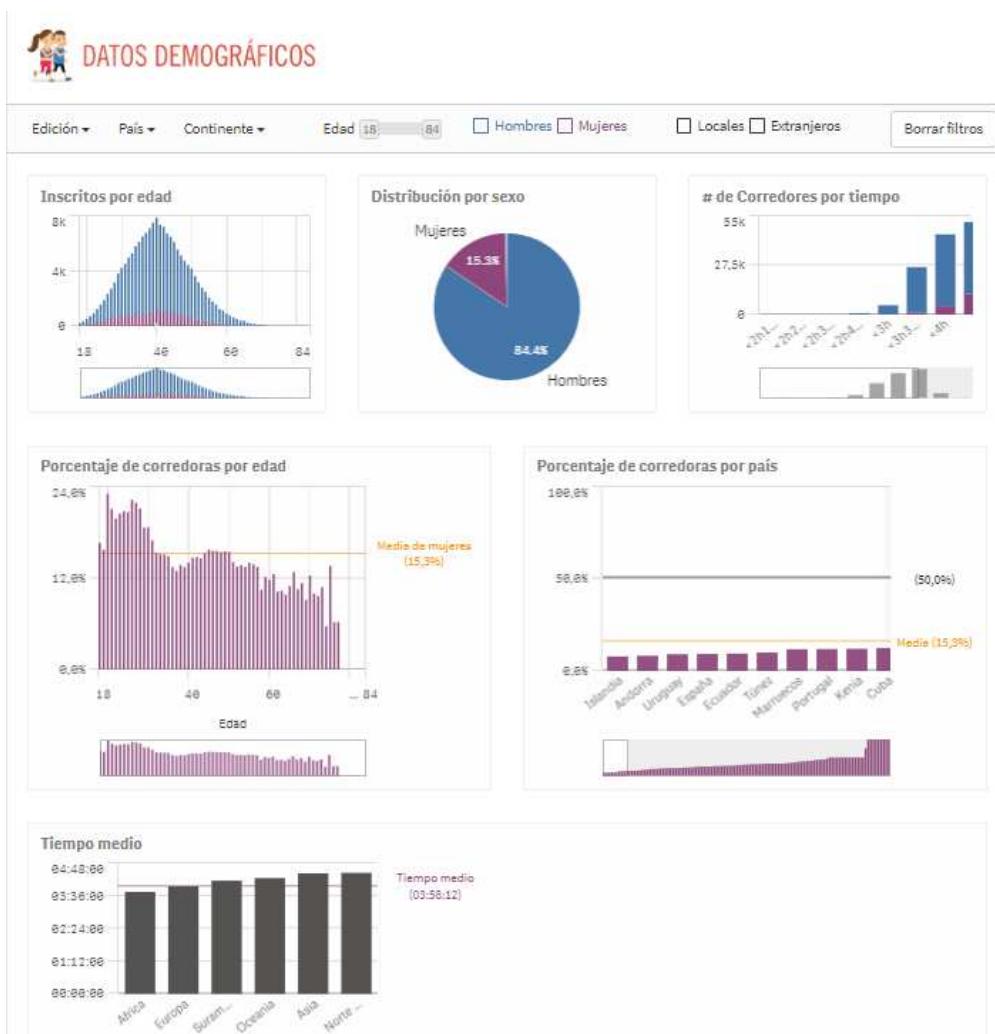


Figura 33. Visualización de datos demográficos con múltiples gráficas combinadas.

Y, por último, a efectos de representar el rendimiento de los maratonianos, también se incluye un gráfico de dispersión que relaciona la edad con el tiempo medio, entre otros indicadores.

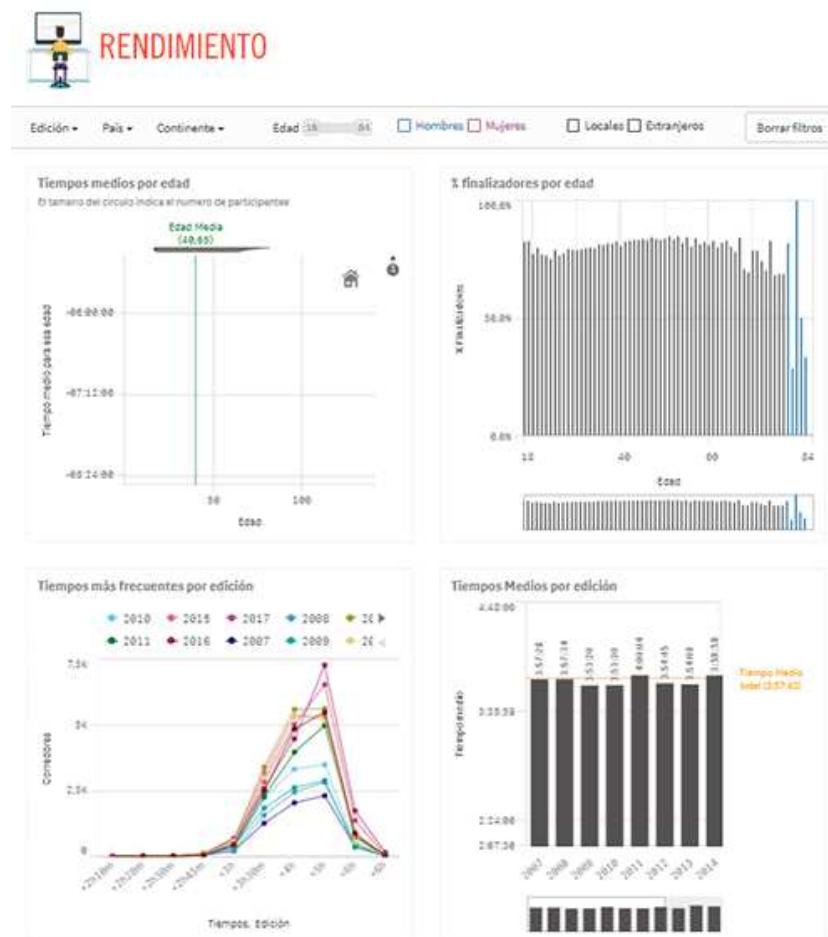
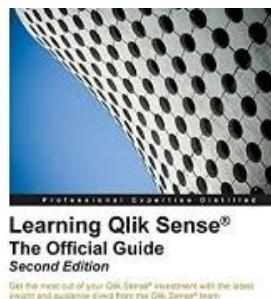


Figura 34. Visualización de datos de rendimiento.

Learning Qlik Sense

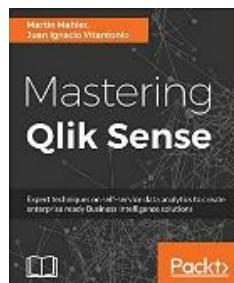
Ilacqua, C. y Cronström, H. (2015). *Learning Qlik Sense: The Official Guide*. Birmingham (UK): Packt Publishing.



Guía oficial para obtener el máximo rendimiento de Qlik Sense.

Mastering Qlik Sense

Mahler, M. y Vitantonio, J. (2018). *Mastering Qlik Sense*. Birmingham (UK): Packt Publishing.



Técnicas avanzadas en análisis de datos para crear soluciones de *business intelligence*.

Onboarding Qlik Sense

Qlik. (S. f.). *Onboarding Qlik Sense*. <https://help.qlik.com/es-ES/onboarding>

Aquí encontrarás una somera guía con cinco sencillos pasos para aterrizar e iniciarse rápidamente en Qlik Sense.



Qlik vídeos

Qlik. (S. f.). *Vídeos*. <https://helpqlik.com/es-ES/videos>

Colección de pequeñas píldoras en formato vídeo para profundizar en los detalles de esta herramienta.



Documentación ▾ Vídeos

Biblioteca de recursos

Qlik. (S. f.). *Biblioteca de recursos.* [https://www.qlik.com/es-es/resource-library?
_ga=2.205942868.615898900.1597238298-1925196475.1594728550](https://www.qlik.com/es-es/resource-library?_ga=2.205942868.615898900.1597238298-1925196475.1594728550)

Porque no todo es programación, en esta dirección tienes una excelente biblioteca de recursos relacionados con la visualización de datos y sus tendencias.

Biblioteca de recursos

- 1.** Qlik Sense es una plataforma:
 - A. Orientada a datos.
 - B. Orientada a la visualización.
 - C. Orientada al descubrimiento de la información.
 - D. Todas las respuestas son válidas.

- 2.** Qlik integra directamente los datos:
 - A. En una visualización.
 - B. En la app.
 - C. En una fuente de datos.
 - D. En un fichero de datos.

- 3.** Qlik Datamarket es:
 - A. Un repositorio de datos, con *datasets* tanto de uso libre como de pago.
 - B. Un mercado disponible para los usuarios que desean adquirir gráficas predefinidas.
 - C. Un entorno abierto en el que los usuarios pueden compartir *datasets*.
 - D. Todas las respuestas son válidas.

- 4.** Una dimensión en Qlik es:
 - A. Un gráfico.
 - B. Un cálculo.
 - C. La característica del dato.
 - D. Una tabla.

5. Una medida en Qlik es:

- A. Un cálculo.
- B. Un gráfico.
- C. La característica del dato.
- D. Una tabla.

6. Cuando distribuimos los datos a lo largo del tiempo, estamos definiendo:

- A. La dimensión de la gráfica.
- B. La medida de la gráfica.
- C. Un filtro de selección.
- D. Ninguna de las opciones es correcta.

7. El gestor de datos nos permite:

- A. Ver la estructura de las tablas y las relaciones existentes entre ellas.
- B. Modificar la estructura de las tablas en el *dataset* de origen
- C. Trabajar con las tablas, pero sin modificar las relaciones.
- D. Gestionar los datos de la cuenta en Qlik Sense.

8. Las tablas pivotantes nos permiten:

- A. Analizar datos por múltiples medidas.
- B. Presentar dimensiones y medidas como filas y columnas de una tabla.
- C. Analizar datos por múltiples dimensiones.
- D. Todas las respuestas son válidas.

9. Las historias de Qlik es el equivalente a:

- A. Un *dashboard*.
- B. Una gráfica.
- C. Una captura de pantalla.
- D. Todas las respuestas anteriores son correctas.

10. Un marcador permite:

- A. Recordar una dimensión que nos ha resultado útil.
- B. Almacenar el resultado de aplicar un filtro.
- C. Aplicar una medida de forma rápida.
- D. Ninguna de las anteriores.

Herramientas de Visualización

Tema 9. Tableau. Introducción y funcionalidades

Índice

[Esquema](#)

[Ideas clave](#)

[9.1. Introducción y objetivos](#)

[9.2. Instalación e interfaz de Tableau](#)

[9.3. Estantes Columnas y Filas. Tarjeta Marcas](#)

[A fondo](#)

[Tableau: Create and Learn](#)

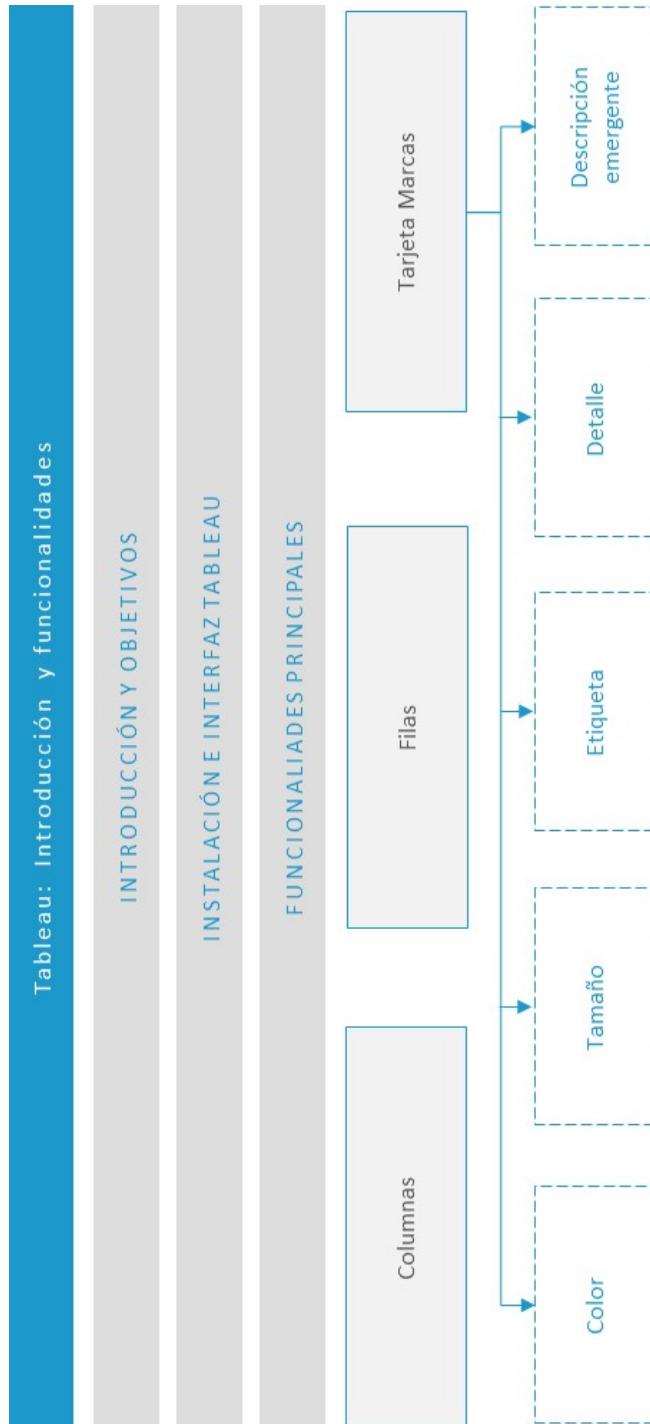
[Visual Data Storytelling with Tableau](#)

[Galería visual de Tableau](#)

[Blog Tableau](#)

[Vídeos de capacitación gratuitos](#)

[Test](#)



9.1. Introducción y objetivos

Tableau es una herramienta de visualización de datos muy potente y visual. Estas dos características, junto sus posibilidades para el análisis integral de los datos, hacen que Tableau ocupe un lugar destacado en el cuadrante mágico de Gartner, junto a Power BI y Qlik.

Accede a la página web oficial de Tableau a través del siguiente enlace:

<https://tableau.com/es-es/>

Descubra cómo Tableau puede ayudarlo a usar los datos para administrar el cambio

Con Tableau, muchas organizaciones son capaces de responder más rápidamente, adaptarse a las nuevas realidades comerciales y capacitar al personal con datos.



Figura 1. Tableau.

Tableau Software es una compañía estadounidense de *software* con sede en Seattle. La compañía fue fundada en el Departamento de Ciencias de la Computación de la Universidad de Stanford entre 1997 y 2002. El profesor Pat Hanrahan y su estudiante de doctorado, Chris Stolte, quien se había especializado en técnicas de visualización para explorar y analizar bases de datos relacionales y cubos de datos, condujeron sus investigaciones hacia el uso de visualizaciones basadas en tablas para explorar bases de datos relacionales multidimensionales.

Juntos crearon un lenguaje de consulta estructurado para bases de datos con un lenguaje descriptivo para renderizar gráficos e inventaron un lenguaje de visualización de base de datos llamado VizQL (Visual Query Language).

VizQL formó el núcleo del sistema Polaris, una interfaz para explorar grandes bases de datos multidimensionales. En 2003, después de que Stolte reclutó a su ex socio comercial y amigo, Christian Chabot, para que tomara el papel de CEO, Tableau se separó de Stanford con una aplicación de *software* homónima. El producto consulta bases de datos relacionales, cubos, base de datos en la nube y hojas de cálculo y luego genera una serie de tipos de gráficos que pueden combinarse en paneles y compartirse a través de una red informática o Internet.

No es necesario que tener conocimientos de programación de algún tipo, todo lo que se necesita son datos para crear con Tableau informes que sean visualmente atrayentes y que cuente una historia que impresione. Con la función de arrastrar y soltar, se pueden crear fácilmente historias o informes utilizando solo el ratón y un poco de imaginación.

En 2019 Salesforce firmó un acuerdo definitivo para la adquisición de Tableau.

9.2. Instalación e interfaz de Tableau

Tableau está disponible en diferentes versiones de uso:

Tableau Online

Es la opción SaaS que se ejecuta en la nube hospedada por el propio Tableau. Permite el trabajo colaborativo desde cualquier lugar, no necesita ningún tipo de configuración ni costes de *hardware*. Es una solución simple completamente hospedada en la nube.

Tableau Server

Permite a las empresas un control completo sobre el *software* de Tableau, pero también implementado en entornos como Amazon Web Services, Google Cloud Platform o Microsoft Azure. Es decir, si se quiere no tener que administrar servidores corporativos, pero a la vez se necesita máximo control sobre el *software* de Tableau y los datos empleados, así como aprovecharse de la escalabilidad de las soluciones de nube pública.

Tableau Server también tiene su entorno *inhouse* por si se desea desplegar las capacidades corporativas de este *software* en las instalaciones físicas de la empresa.

Tableau Desktop

Es la solución instalable en los equipos de oficina. Permite garantizar la información confidencial dentro de su propio *firewall* corporativo o equipo autónomo. Es la versión que emplearemos para el desarrollo de los casos en la asignatura.

Tecnología avanzada en memoria: el motor de datos

La mayor parte del *software* de análisis de datos ofrece muchas características sofisticadas, pero casi todas fallan cuando se trata de operar con grandes cantidades de datos, aquí es donde la tecnología avanzada en memoria de Tableau es fundamental para todos aquellos que necesitan obtener informes de cada vez mayores cantidades de datos.

El motor de datos de Tableau es una revolucionaria **base de datos analítica** en memoria, diseñada para superar las limitaciones de las bases de datos y de los silos de datos existentes. Es capaz de ejecutarse en computadoras de prestaciones normales, porque aprovecha la jerarquía de memoria completa del disco a la caché L1. Cambia la curva entre los grandes datos y el análisis rápido, el análisis *ad hoc* de datos masivos tiene lugar en segundos y no se requiere un modelo de datos fijo.

Interfaz Tableau

Respecto a la interfaz de Tableau, esta presenta funcionalmente los mismos elementos que otros productos y herramientas de visualización.

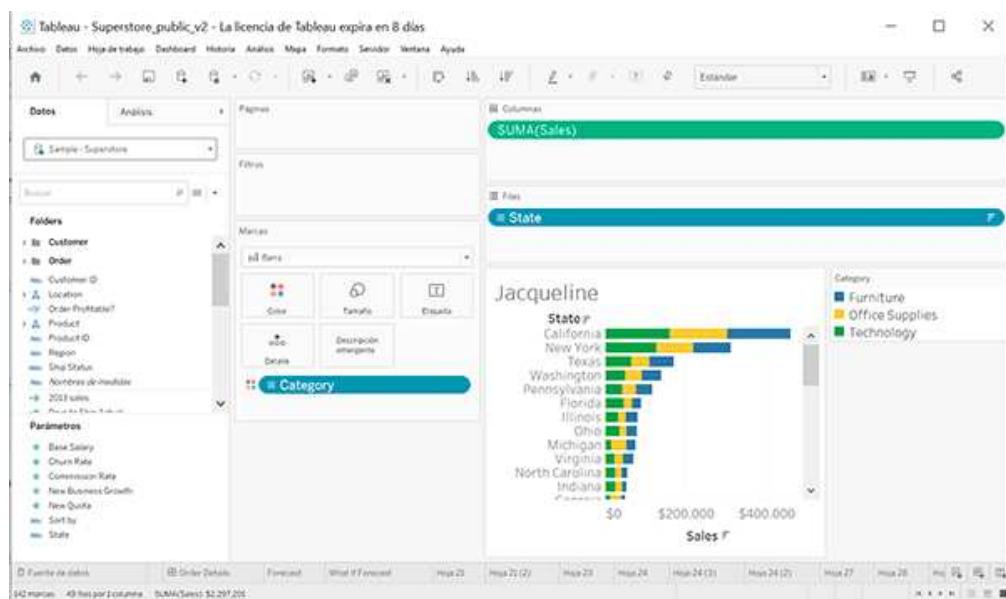


Figura 2. Interfaz Tableau.

Dispone de una **barra de herramientas superior** donde podemos encontrar opciones de menú generales como Archivo, que permite las operaciones habituales de Abrir, Guardar, Importar/Exportar informes o Exportar la visualización a PDF, entre otras muchas opciones. El resto de opciones del menú, tales como Datos, Hoja de trabajo o Dashboard incorporan elementos específicos que nos permitirán crear nuestras visualizaciones, aunque en su mayor parte se pueden emplear a través de las opciones de los menús laterales.

Existe un **menú lateral izquierdo**, a través del cual podremos acceder a los campos y datos de nuestro *dataset*. En terminología Tableau:

- ▶ Los datos descriptivos se llaman dimensiones.
- ▶ Los datos numéricos se denominan medidas.

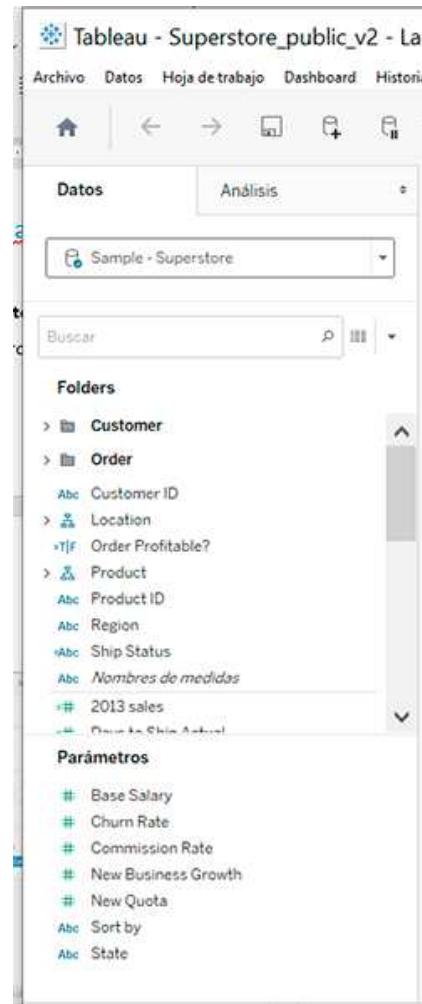


Figura 3. Dimensiones y medidas.

Arrastrando al panel tanto las medidas como las dimensiones, podremos empezar a crear nuestras visualizaciones. Cuando se combinan, ayudan a visualizar el rendimiento de los datos dimensionales con respecto a los datos que son medidas. Estos conceptos los iremos desarrollando con la práctica.

En la **parte superior de la pantalla**, justo encima del panel de visualización, se sitúan los dos elementos más importantes del interfaz: el espacio de **Columnas** y el espacio de **Filas**. Arrastrando los campos de datos (dimensiones o medidas) a estos estantes, podremos empezar a crear las visualizaciones.

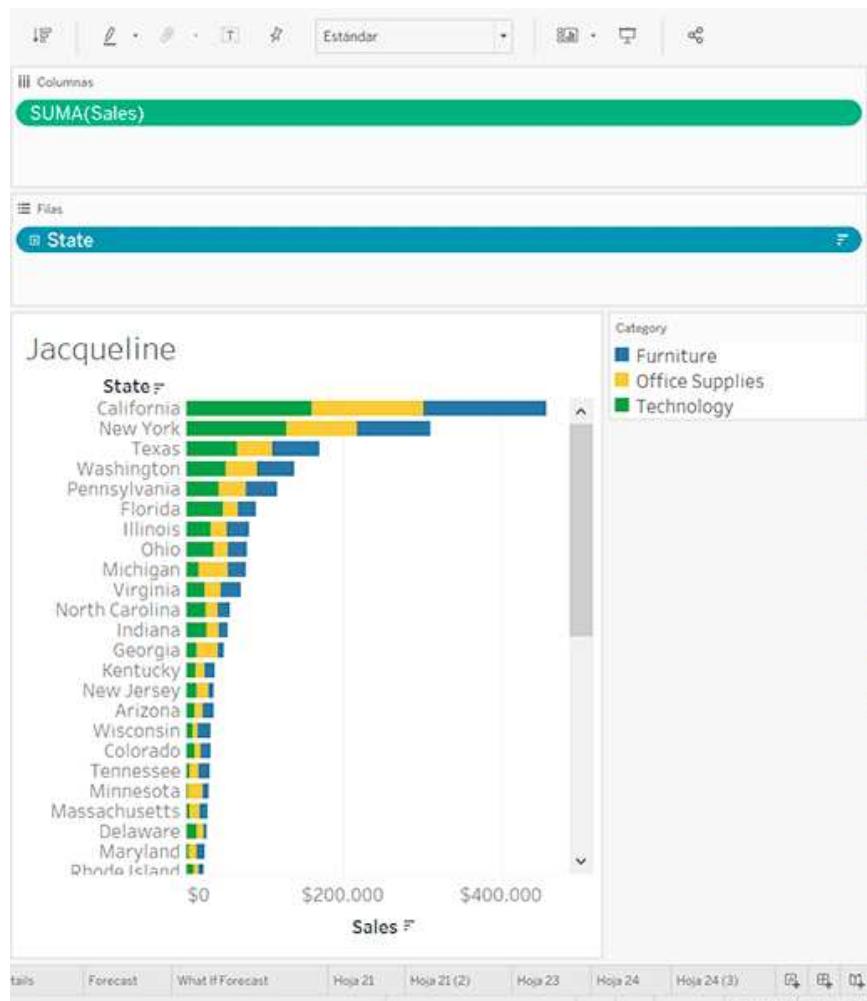


Figura 4. Estantes Columnas y Filas.

En la **parte inferior de la pantalla**, existe la opción de agregar nuevas hojas a la visualización, como si fueran pestañas de una hoja Excel. También incorpora una primera pestaña denominada **Fuente de Datos**, donde podremos visualizar la estructura de datos del dataset, ver los valores que contiene y realizar transformaciones de datos como **Crear campos calculados**.

A continuación del menú lateral izquierdo se sitúa un bloque de opciones bajo los epígrafes de Páginas, Filtros y Marcas.

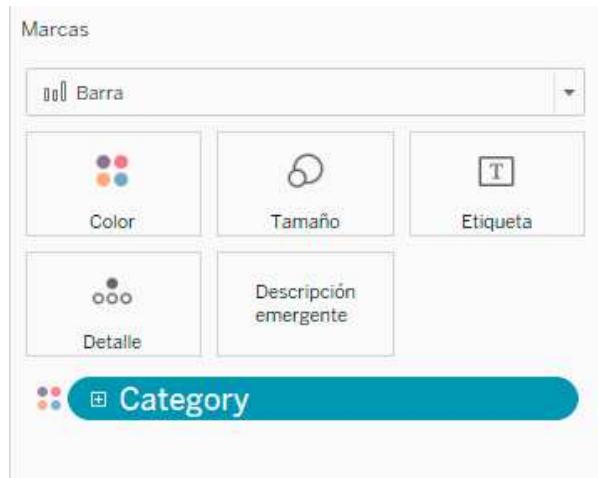


Figura 5. Marcas.

- ▶ El bloque o tarjeta **Páginas** permite dividir una vista en una serie de páginas para que se pueda analizar mejor cómo un campo específico afecta al resto de los datos en una vista. Es útil, por ejemplo, cuando se quiere ver cómo evolucionan los datos con el paso del tiempo, y se pueden visualizar todas las páginas de forma automática como si se tratase de una Historia, o bien seleccionar una página concreta.
- ▶ El bloque de **Filtro** permite definir qué datos excluir e incluir en la visualización. Seleccionando los contenedores de datos concretos, se pueden crear vistas a medida.
- ▶ El bloque de **Marcas** tiene el conjunto de funciones más potentes y simples de utilizar en Tableau. Son intuitivas y con gran flexibilidad, ya que permiten estructurar los datos, visualizarlos, darles forma y añadir dimensiones con solo arrastrar los campos ubicados en el lateral izquierdo a las opciones de **Color**, **Tamaño**, **Etiqueta**, **Detalle** o **Descripción emergente**.

9.3. Estantes Columnas y Filas. Tarjeta Marcas

Vamos a recorrer de forma muy práctica diferentes casos de uso en la creación de visualizaciones con Tableau, y veremos cómo utilizar las diferentes tarjetas y opciones. Para ello emplearemos un *dataset* con información sobre ventas e iremos utilizando paso a paso, de forma incremental, nuevas opciones de la herramienta.

Supongamos que queremos crear un simple diagrama de barras para mostrar los beneficios obtenidos por cada categoría de productos.

Arrastramos el campo Category al estante Columnas y el campo Profit al estante Filas. Si además queremos que la gráfica resultante diferencie en cada categoría las diferentes subcategorías de productos que la componen, arrastramos el campo Sub-Category a la opción Color ubicada en la tarjeta Marcas.

El resultado obtenido es el siguiente:

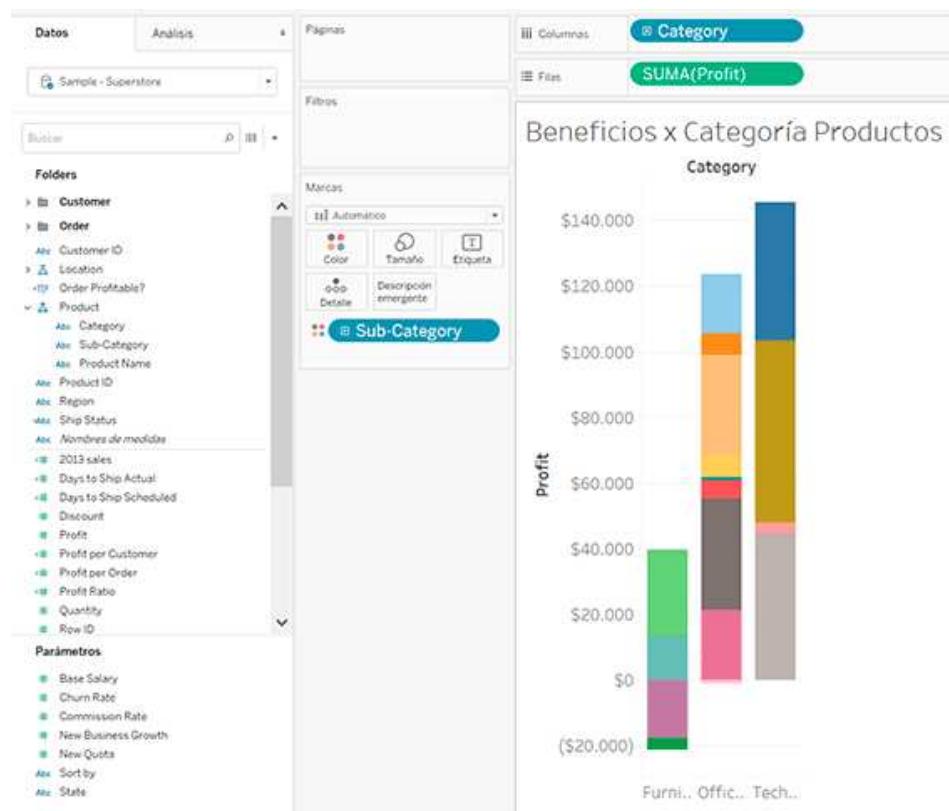


Figura 6. Beneficios agrupados por categorías de productos.

Por defecto, Tableau estima cuál es el mejor tipo de representación a aplicar en cada caso a los datos. En este ejemplo, el tipo de gráfica Automática se corresponde con un gráfico de barras, no obstante también podemos escoger manualmente la visualización que queremos aplicar.

En la tarjeta Marcas existe un desplegable con todas las opciones de gráficas disponibles. Escogemos el tipo Círculo y volvemos a quitar el campo Sub-Category de la opción Color. El resultado es el siguiente:

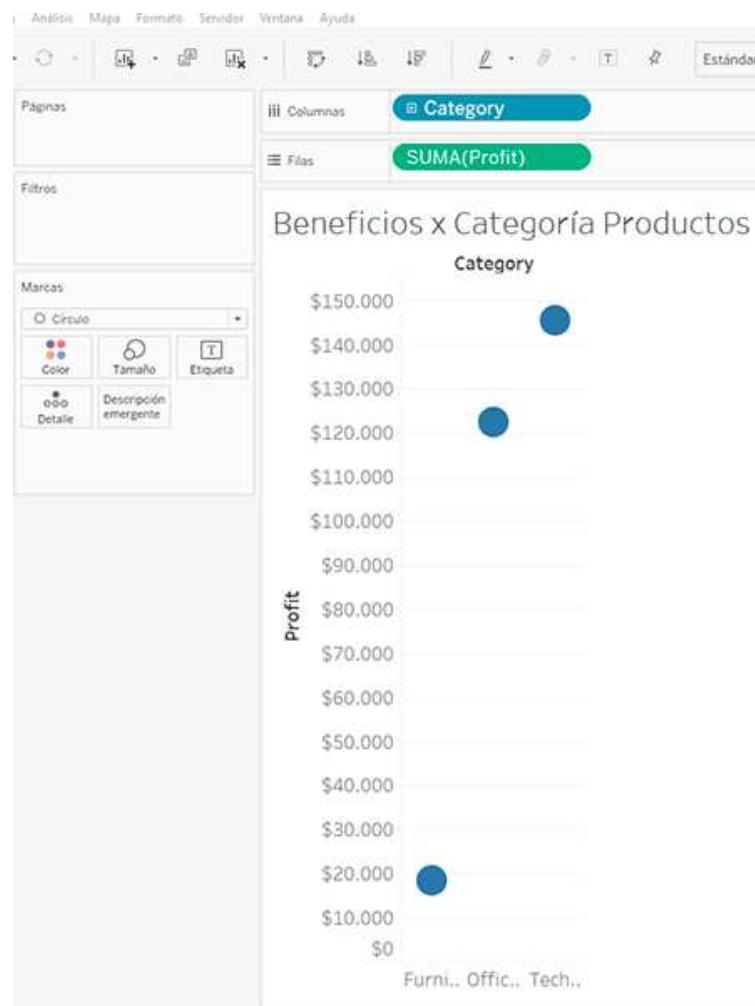


Figura 7. Distribución de los beneficios en un diagrama de dispersión.

En los estantes podemos incluir todos los campos que queramos. Si al estante Columnas le añadimos el campo Sub-Category, forzamos a que la gráfica nos muestre los datos ordenados primero en categorías y después en subcategorías.

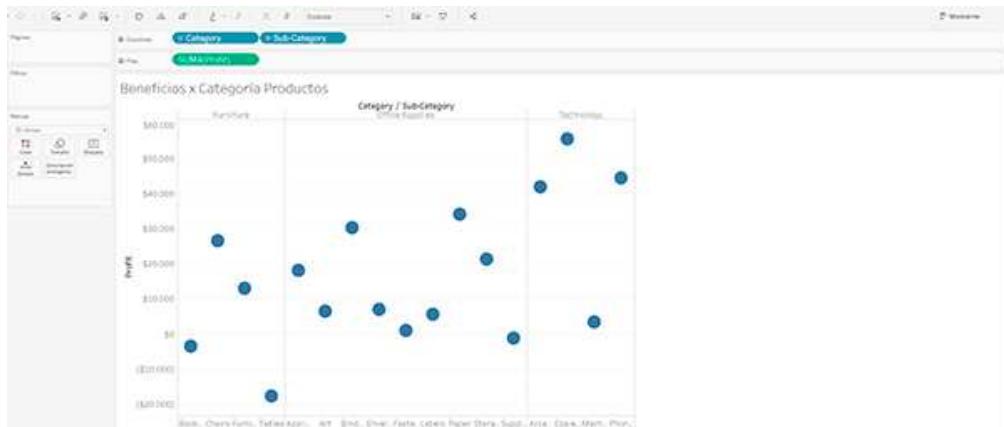


Figura 8. Distribución de los beneficios por categorías y subcategorías.

Pasando el ratón por encima de cualquiera de los círculos representados en la gráfica anterior, obtendremos **información de detalle** sobre lo que representa y sus datos.

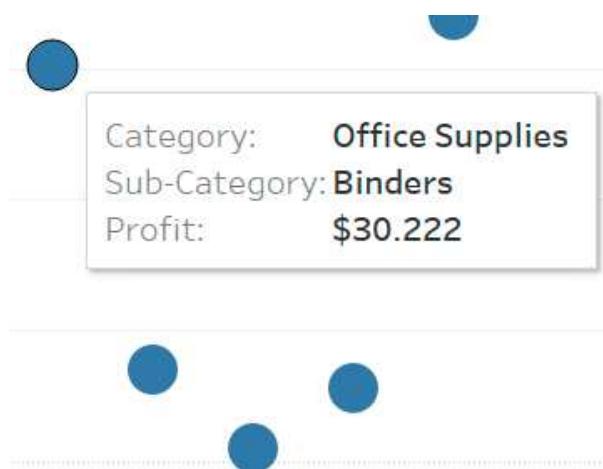


Figura 9. Etiqueta automática con información sobre el punto.

Si además queremos etiquetar cada círculo con la subcategoría de producto que representa, arrastraremos el campo Sub-Category a la opción **Etiqueta** de la tarjeta Marcas.

Ideas clave

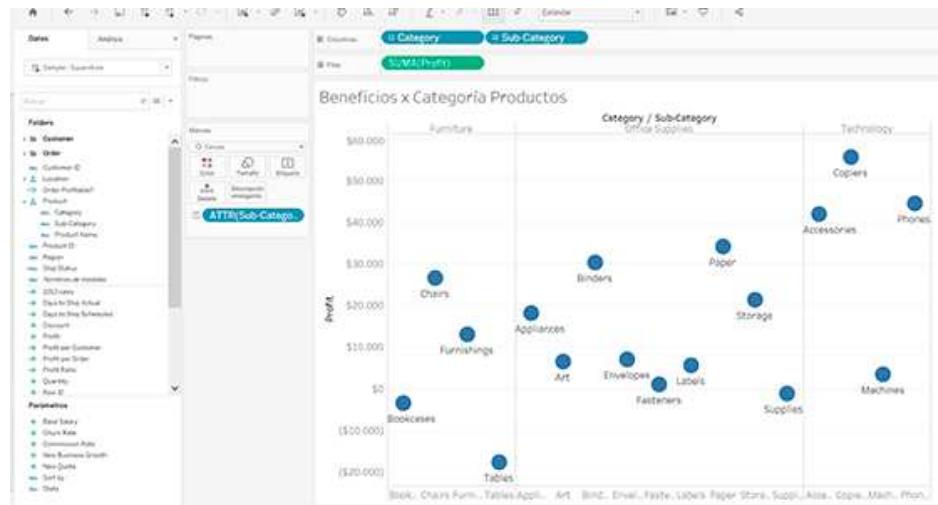


Figura 10. Visualización con etiquetas en cada subproducto.

Por último, podemos **colorear** todos los puntos en función de la categoría de producto que representan. Para ello arrastramos el campo Category a la opción Color de la tarjeta Marcas.

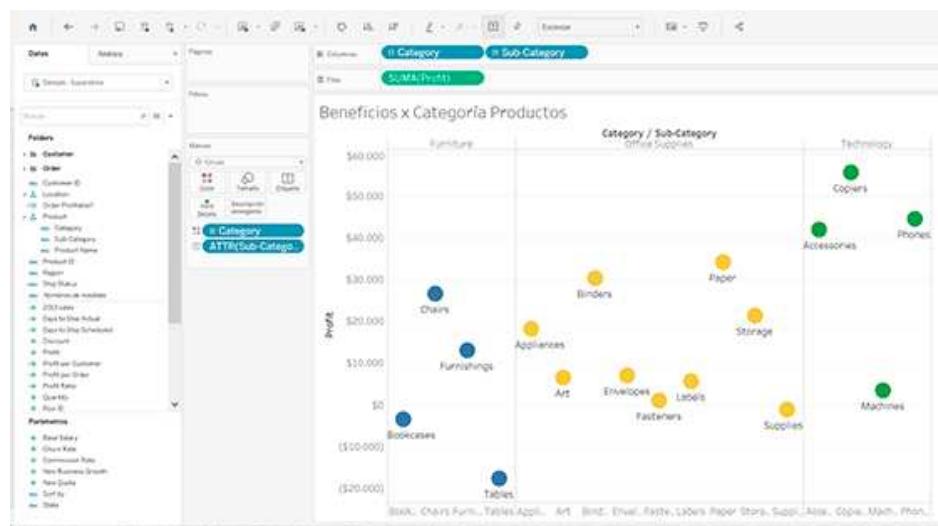


Figura 11. Visualización de las categorías por colores.

Existe otra opción también muy utilizada en la tarjeta de Marcas denominada **Tamaño**. Si arrastramos el campo Benefit a Tamaño, cada círculo tendrá un tamaño proporcional al beneficio obtenido con su venta.



Figura 12. Tamaño de los círculos proporcional a los beneficios.

Como habéis podido comprobar, Tableau nos permite crear potentes visualizaciones con simples desplazamientos de los datos a la tarjeta de Marcas y a los estantes de Columnas y Filas.

Nos hemos aproximado de forma general a la funcionalidad de las opciones de la tarjeta Marcas. A continuación, vamos a entrar en el detalle de cada una de ellas.

Color

La función principal de esta opción es la de discriminar por colores el contenido del gráfico. El criterio empleado para dicha distinción se basa en el campo o campos arrastrados a la opción.

Ahora bien, Tableau aplica por defecto una paleta de colores y selecciona de ella los colores utilizados. Esta situación se puede **personalizar** y el usuario tiene la posibilidad de seleccionar el color aplicado a cada elemento de datos, así como la paleta.

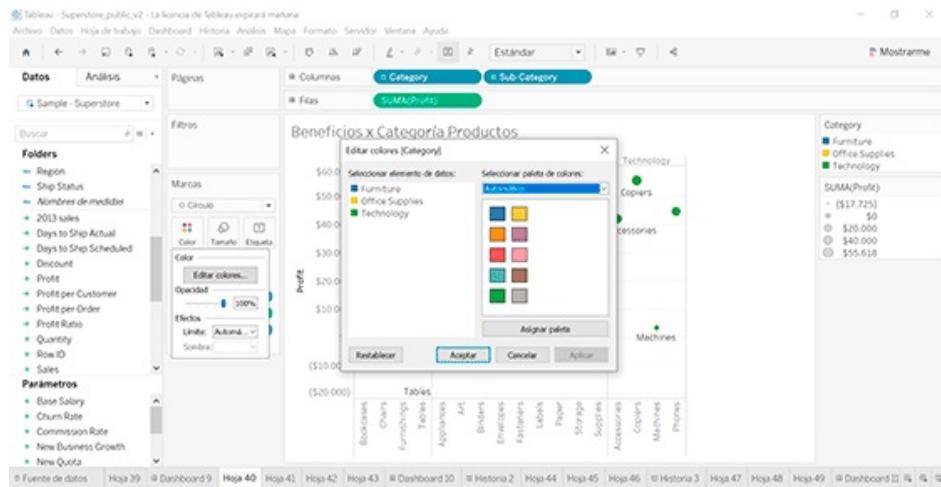


Figura 13. Editar colores.

Tamaño

Esta opción nos permite modificar a discreción el tamaño general de los objetos. Es muy útil para **destacar** los objetos de la visualización y darles una proporcionalidad adecuada.

Etiqueta

Esta opción no solo asigna la etiqueta a los objetos de la visualización, sino que también permite **especificar algunos criterios de visualización** como, por ejemplo, mostrar la etiqueta a todos los elementos (por defecto), solo a los elementos extremos (máx. y min.) o a una selección de objetos.

También podemos personalizar el contenido de la etiqueta.

Detalle

Esta opción es muy útil porque nos permite incluir nuevas **vistas a la visualización**, sin que formen parte de los estantes Columna o Fila. Arrastrando un campo a la opción Detalle, incorporamos el nivel de detalle marcado por el campo. No incluye opciones adicionales.

Descripción emergente

Al pinchar en esta opción, aparece un **editor de texto** que nos permite editar el texto emergente o *tooltip* que aparece al pasar el ratón por las barras, círculos, etc. de nuestra visualización.

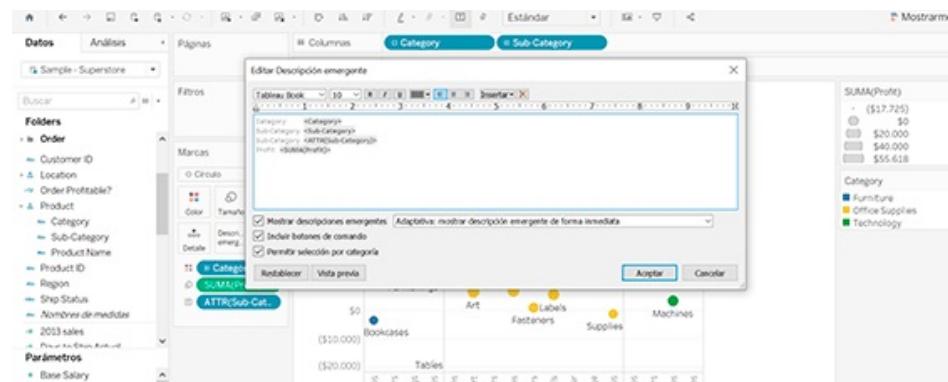


Figura 14. Editar descripción emergente.

Por defecto, el *tooltip* incorpora los campos que hemos empleado en los estantes y en la tarjeta de Marcas.

Con este editor de texto y, en especial, gracias al desplegable Insertar que nos permite incluir el campo del dataset que deseemos en la descripción emergente, podremos personalizarla a nuestra elección.

En nuestro caso, supongamos que solo queremos visualizar la subcategoría y el beneficio. Editamos el texto y el resultado es el siguiente:

Ideas clave

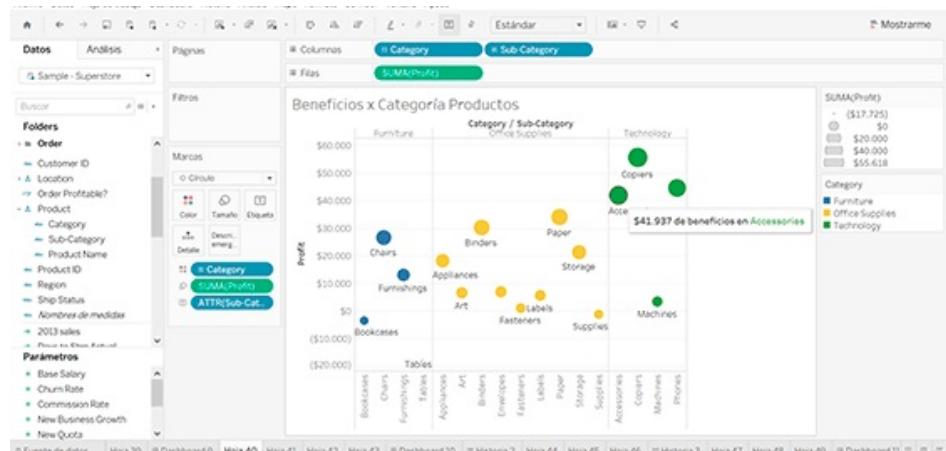


Figura 15. Descripción emergente personalizada.

Tableau: Create and Learn

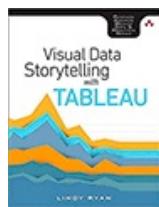
Silva, R. F. (2019). *Tableau: Create and Learn*. Amazon.



Este libro con más de 300 ilustraciones recorre los aspectos más relevantes de Tableau, desde su instalación hasta la creación de dashboards.

Visual Data Storytelling with Tableau

Ryan, L. (2018). *Tableau: Create and Learn*. Amazon.



Si quieres contar una historia de datos con Tableau, te recomendamos este libro.

Visual Data Storytelling with Tableau te ayuda a organizar los datos, estructurar el análisis, profundizar en la exploración y crear visualizaciones avanzadas, pero siempre teniendo en mente que el objetivo final es contar una historia basada en datos.

Galería visual de Tableau

Tableau Software. (S. f.). *Galería visual de Tableau*. <https://www.tableau.com/es-es/solutions/gallery>

En la siguiente galería podemos ver ejemplos de visualizaciones con Tableau.



Blog Tableau

Tableau Public. (S. f.). *Blog: Comunicaciones habituales del equipo de Tableau Public.*

<https://public.tableau.com/s/blog>

En este blog podemos seguir las comunicaciones habituales del equipo de Tableau.



Blog

Vídeos de capacitación gratuitos

Tableau Software. (S. f.). *Vídeos de capacitación gratuitos.*

<https://www.tableau.com/es-es/learn/training/20202>

En este apartado podemos recurrir a pequeñas píldoras oficiales de Tableau.

APRENDIZAJE

Videos de capacitación gratuitos

1. Tableau es una herramienta que permite la visualización de datos:

 - A. Correcto. Aunque no incorpora capacidades para el tratamiento y transformación de los datos.
 - B. Incorrecto. Solo la versión Desktop permite la visualización de datos.
 - C. Incorrecto. Solo la versión SaaS permite la visualización de datos.
 - D. Correcto. Adicionalmente incorpora capacidades avanzadas para el tratamiento y transformación de los datos.
2. Tableau permite el intercambio rápido entre la visualización de datos y la visualización de gráficas:

 - A. Correcto. Desde el menú inferior izquierdo es posible intercambiar dichas vistas.
 - B. Correcto. Pulsando F5.
 - C. Incorrecto. Una vez cargados los datos, ya solo nos movemos en su visualización.
 - D. Incorrecto. Existe la opción de carga de datos y visualización posterior en modo gráfica. La visualización de los datos no es necesaria en el 80 % de las ocasiones.
3. ¿Dónde se ubican las opciones de asignación de colores y tamaños de los objetos gráficos?

 - A. En el menú lateral izquierdo dentro del bloque de Marcas.
 - B. No existen dichos menús.
 - C. No existe el menú de asignación de tamaños.
 - D. No existe el menú de asignación de colores.

4. ¿Qué fases ordenadas incorpora un proyecto de análisis y visualización de datos?

 - A. Preparación – Obtención - Modelado – Visualización – Reporte de datos.
 - B. Obtención – Modelado – Preparación – Modelado – Visualización – Reporte de datos.
 - C. Obtención – Preparación – Modelado – Visualización – Reporte de datos.
 - D. Obtención – Visualización de datos.
5. Los campos de los ejes y de los valores son definidos automáticamente por Tableau:

 - A. Correcto. Así nos facilita la experiencia como usuarios.
 - B. Incorrecto. Son definidos por el usuario.
 - C. Correcto y no se pueden modificar.
 - D. Ninguna de las anteriores.
6. Las magnitudes principales que representan los ejes se definen:

 - A. En la opción Detalle de la tarjeta Marcas.
 - B. En los estantes de Filas y Columnas.
 - C. En la opción Ejes de la tarjeta Marcas.
 - D. Ninguna de las anteriores. Tableau selecciona las magnitudes oportunas en cada momento.
7. El tipo de visualización se puede modificar:

 - A. En la tarjeta Páginas.
 - B. En la tarjeta Marcas.
 - C. En la opción Detalle de la tarjeta Marcas.
 - D. Ninguna de las anteriores. Tableau autoselecciona el tipo de gráfica y no es modificable.

- 8.** En una gráfica de dispersión, el tamaño de los círculos de la gráfica será siempre proporcional al valor del dato que representa:
- A. Correcto.
 - B. No necesariamente, salvo que depositemos en la opción Tamaño de la tarjeta *Marcas* el campo que queremos vincular al tamaño.
 - C. No es una opción existente en Tableau.
 - D. Incorrecto. No es posible modificar el tamaño de los elementos de una gráfica.
- 9.** La opción de Descripción emergente permite definir el contenido de un *tooltip*.
- A. Incorrecto. Los *tooltip* son contenidos fijos no modificables.
 - B. Correcto. Desde dicha opción podemos editar su contenido.
 - C. Incorrecto. Los *tooltip* se pueden modificar pero haciendo clic en ellos.
 - D. Incorrecto. Los *tooltip* se modifican pero desde la opción de *dashboard*.
- 10.** La opción de Etiqueta permite definir el contenido de las etiquetas en los objetos de la visualización:
- A. Incorrecto. Las etiquetas son contenidos fijos no modificables.
 - B. Correcto. Desde dicha opción podemos editar su contenido.
 - C. Incorrecto. Las etiquetas se pueden modificar pero haciendo clic en ellas.
 - D. Incorrecto. Las etiquetas se modifican pero desde la opción de *dashboard*.

Herramientas de Visualización

Tema 10. Tableau. Filtros, páginas, dashboards e historias

Índice

Esquema

A fondo

Practical Tableau: 100 Tips, Tutorials, and Strategies from a Tableau Zen Master

Visual Analytics with Tableau

KPI – Ejemplos prácticos

Prácticas recomendadas para ordenar los datos con Tableau Prep

Ideas clave

10.1. Filtros

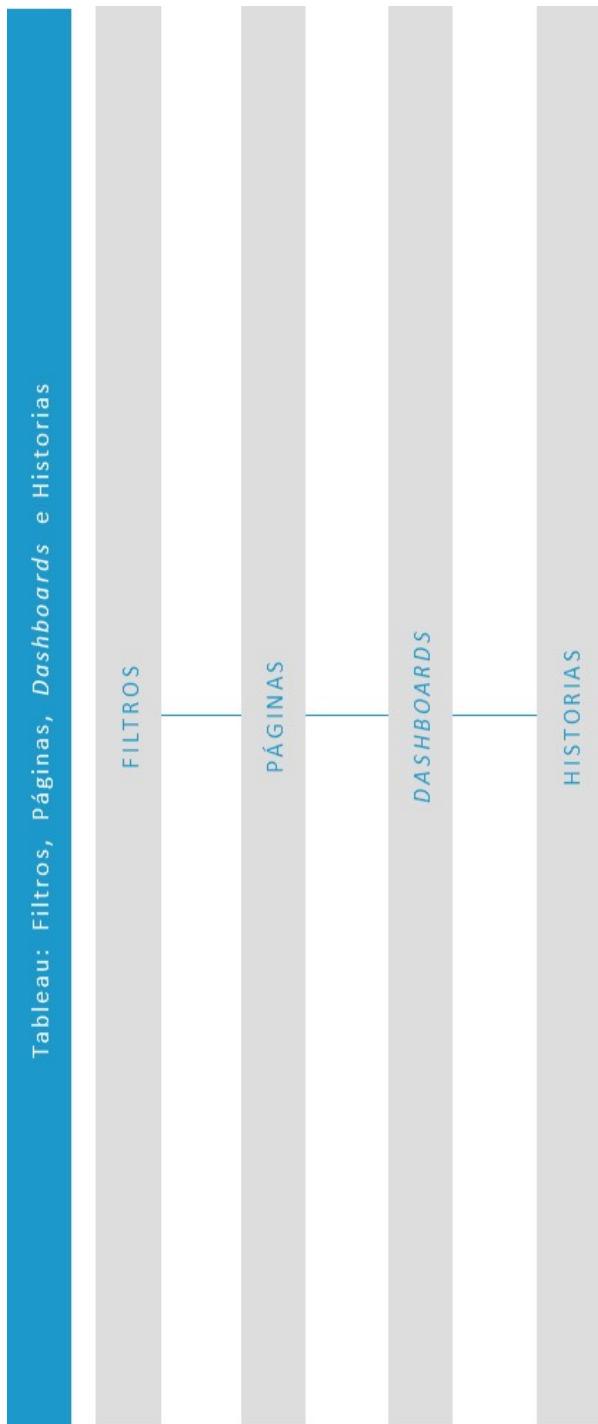
10.2. Páginas

10.3. Dashboards

10.4. Historias

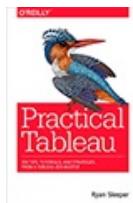
Test

Esquema



Practical Tableau: 100 Tips, Tutorials, and Strategies from a Tableau Zen Master

Sleeper, R. (2018). Practical Tableau: 100 Tips, Tutorials, and Strategies from a Tableau Zen Master. O'Reilly.



Este libro es para todos los niveles y proporciona consejos prácticos para crear visualizaciones de datos interactivas y facilitar la búsqueda de información relevante en ellos.

Visual Analytics with Tableau

Loth, A. (2019). Visual Analytics with Tableau. Wiley.



Este libro hace un completo recorrido de las principales funciones y usos de Tableau, orientado a usuarios con un perfil de negocio no técnico.

KPI – Ejemplos prácticos

Tableau Software. (S. f.). *KPI – Ejemplos prácticos*. <https://www.tableau.com/es-es/learn/articles/kpi-examples>

Existen muchos ejemplos de indicadores de negocio y cómo se trasladan a un cuadro de mando. A continuación, tienes a tu disposición ejemplos de indicadores de facturación, de reducción de costes, de mejora de tiempos, de aumento de la calidad en procesos y de satisfacción de cliente. Una auténtica colección de ejemplos valiosos.

KPI – Ejemplos prácticos

Prácticas recomendadas para ordenar los datos con Tableau Prep

Tableau Software. (S. f.). *Prácticas recomendadas para ordenar los datos con Tableau Prep.* <https://www.tableau.com/es-es/learn/whitepapers/data-prep-best-practices>

La preparación de datos es el proceso de limpiar los datos, cambiar las estructuras incorrectas y combinar varios conjuntos de datos para el análisis. Implica transformar la estructura de los datos, como filas y columnas, y limpiar elementos como tipos de datos y valores.



10.1. Filtros

A la hora de analizar los datos y buscar información relevante en los mismos, las visualizaciones son un elemento fundamental que nos permite su exploración gráfica. Sin embargo, debido al volumen de datos que manejamos y a las dimensiones y complejidad del análisis, en ocasiones la visualización no es suficiente y nos tenemos que apoyar en otras capacidades que nos ofrecen Tableau, como es el caso de los Filtros.

En este tema continuaremos con el mismo *dataset* con información de ventas empleado temas anteriores e iremos incorporando, paso a paso, nuevas funcionalidades de Tableau orientadas a una mejor interpretación de los datos.

Por ejemplo, continuando con el caso anterior, queremos filtrar los beneficios obtenidos en las diferentes categorías de productos, pero con base en unas **subcategorías específicas**: sillas, papel y sobres.

Arrastramos el campo Sub-Category a la tarjeta Filtros y lo mostramos. Aparecerá en la parte derecha de la pantalla, listo para ser usado.

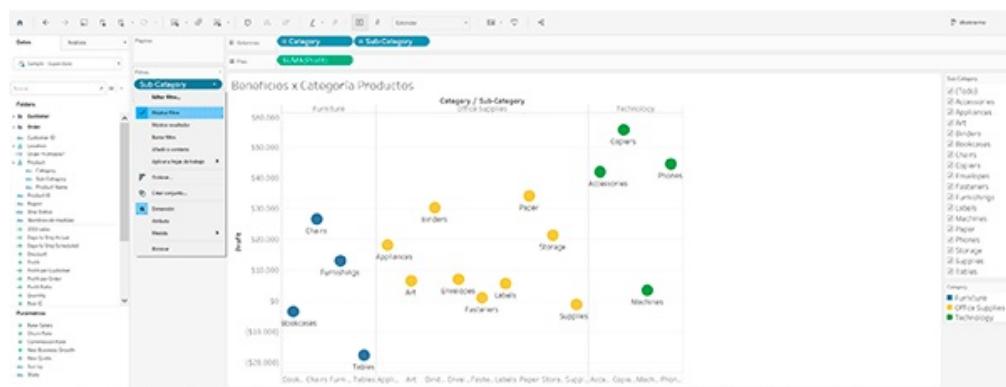


Figura 1. Activación de filtros.

Ya solo queda marcar o desmarcar los valores en el cajetín del filtro. En nuestro caso, habíamos dicho que queríamos ver los beneficios de tres subcategorías de productos: sillas, papel y sobres.

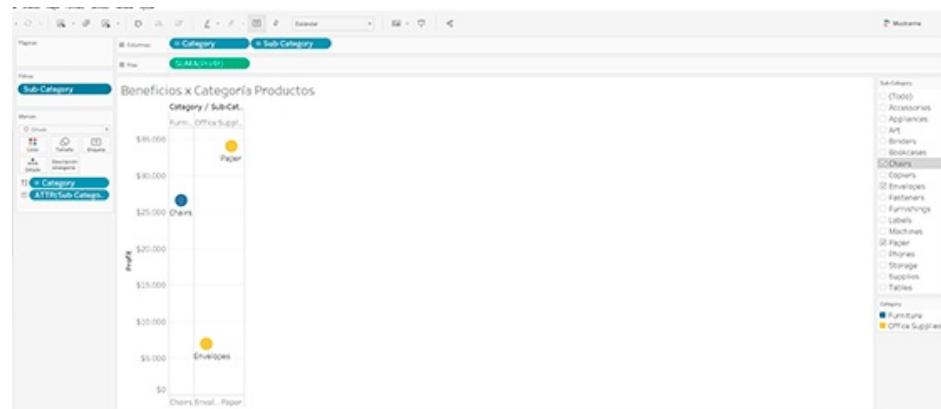


Figura 2. Selección de los beneficios de 3 subcategorías.

En este caso hemos empleado para el filtro todos los valores diferentes que tiene el campo Subcategoría o Sub-Category. Pero también podemos decidir emplear para el filtro únicamente una selección de estos valores.

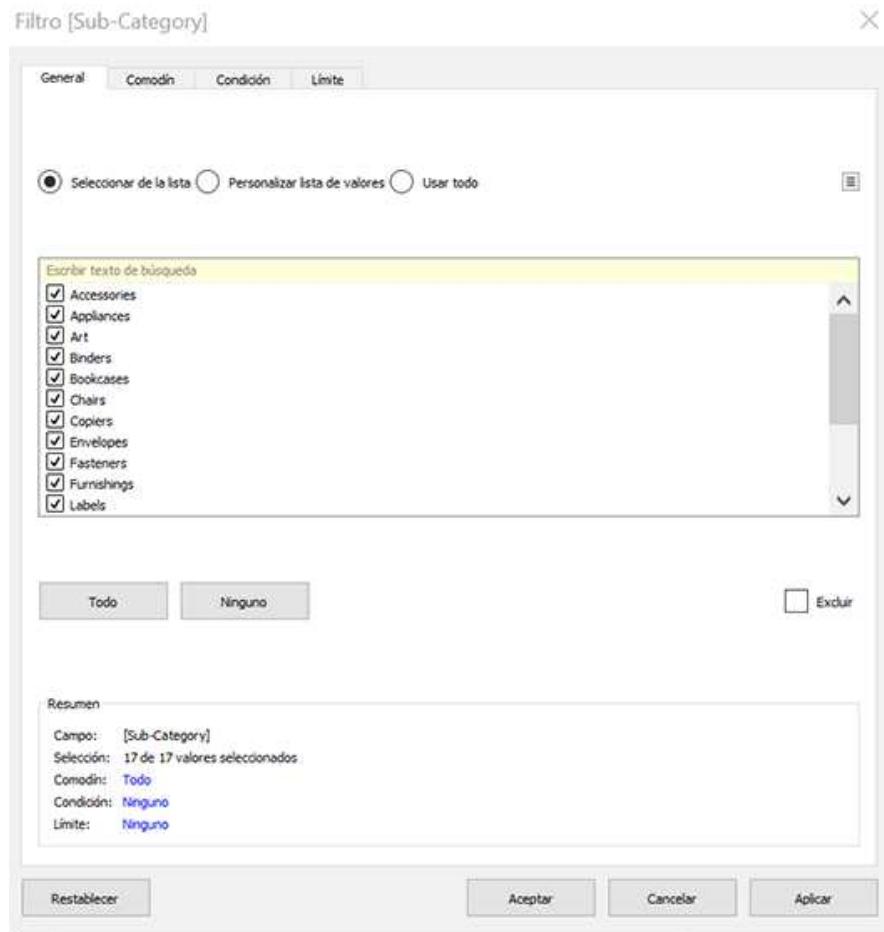


Figura 3. Personalización de los campos de un filtro.

También existe la opción de crear nosotros mismos una **condición de filtro** con base en umbrales o condiciones que deba cumplir. Podemos definir un rango de valores para el campo, crear operaciones matemáticas o directamente aplicar un límite a los datos. Las opciones son muchas y demuestran la flexibilidad de esta opción de filtros.

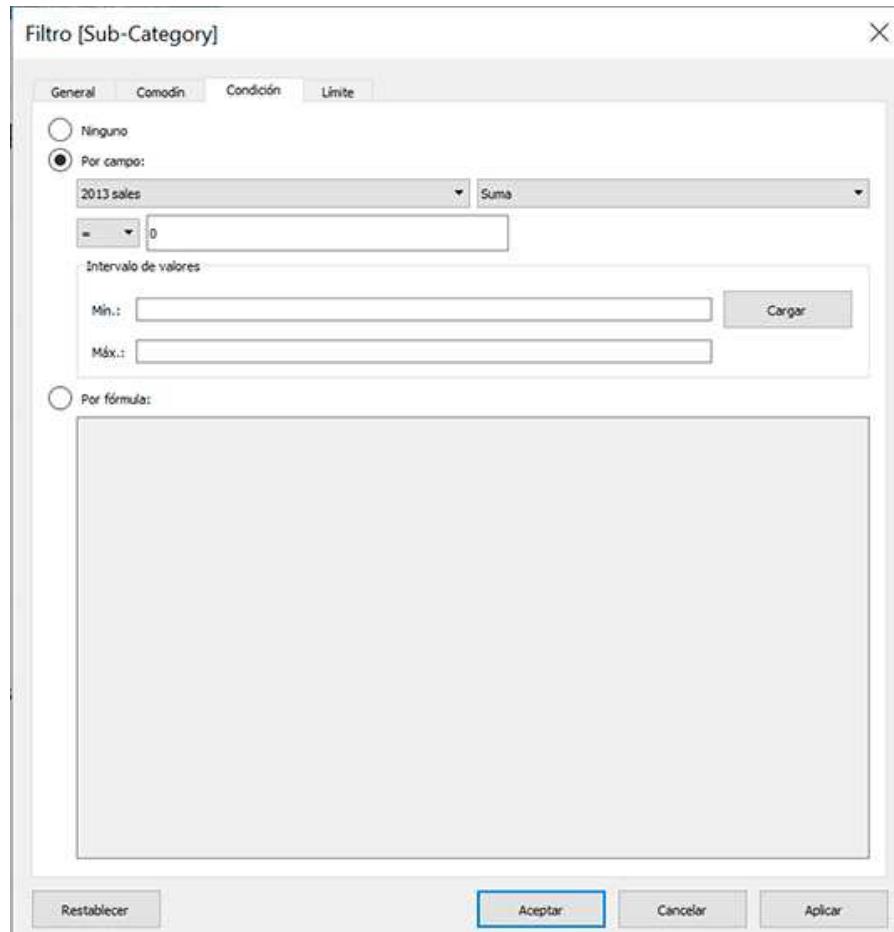


Figura 4. Condiciones de un filtro.

En todo caso, para activar un filtro una vez definido y que aparezca en el lateral derecho listo para su uso, debemos hacer clic en la opción Mostrar filtro.

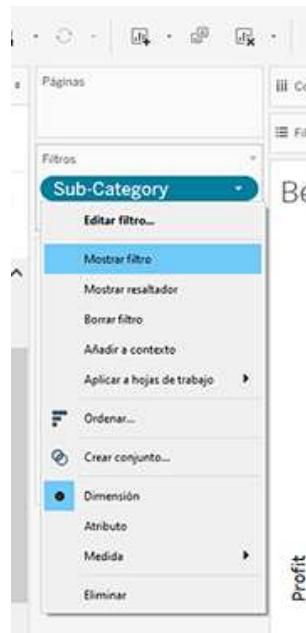


Figura 5. Activación de un filtro.

10.2. Páginas

La tarjeta Páginas nos permite recorrer la visualización y **observar los cambios** en la misma cuando cambiamos los valores de un campo manteniendo el resto fijos.

La vista de una visualización se puede dividir en una serie de páginas con el objetivo de ver **cómo un campo específico afecta al resto de los datos** de dicha vista. Por ejemplo, las páginas se suelen utilizar para mostrar cambios con el paso del tiempo, cada página puede representar un año, un mes o la unidad de medida que definamos.

Veamos en nuestro caso de uso qué es lo que tenemos que hacer para ver la evolución de las ventas a lo largo de los años utilizando las páginas. En el *dataset* que estamos utilizando existe el campo Order date que indica la fecha en que se ha realizado cada pedido. Arrastramos dicho campo a la **tarjeta Páginas**.

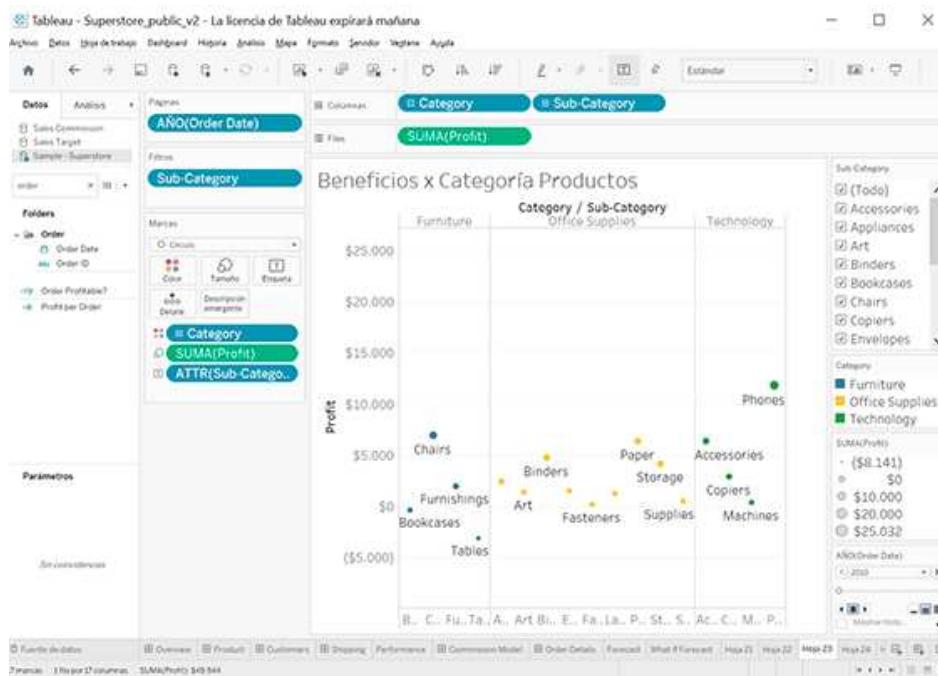


Figura 6. Activación de las Páginas.

También podemos observar cómo en el lateral derecho ahora se muestra un pequeño **panel de control** asociado al campo Order date que hemos seleccionado para las páginas.

En este panel de control podemos recorrer manualmente cada uno de los valores diferentes que tiene este campo. También podemos darle al **play** y Tableau mostrará secuencialmente la visualización de la gráfica, personalizando los beneficios por categoría y subcategoría para los diferentes años. Es decir, los controles de este panel te permiten hojear las páginas automáticamente o pasar a una concreta.

En el *dataset* tenemos guardados los pedidos de los años 2013 a 206. Por lo tanto, se generarán cuatro vistas diferentes. De esta forma nos resulta más sencillo analizar las ventas y podemos ver que a lo largo de estos años (2013-2016) todas las ventas han subido en general pero que en especial, a lo largo de este periodo las ventas de tecnología han despuntado sobre las demás con gran ventaja.

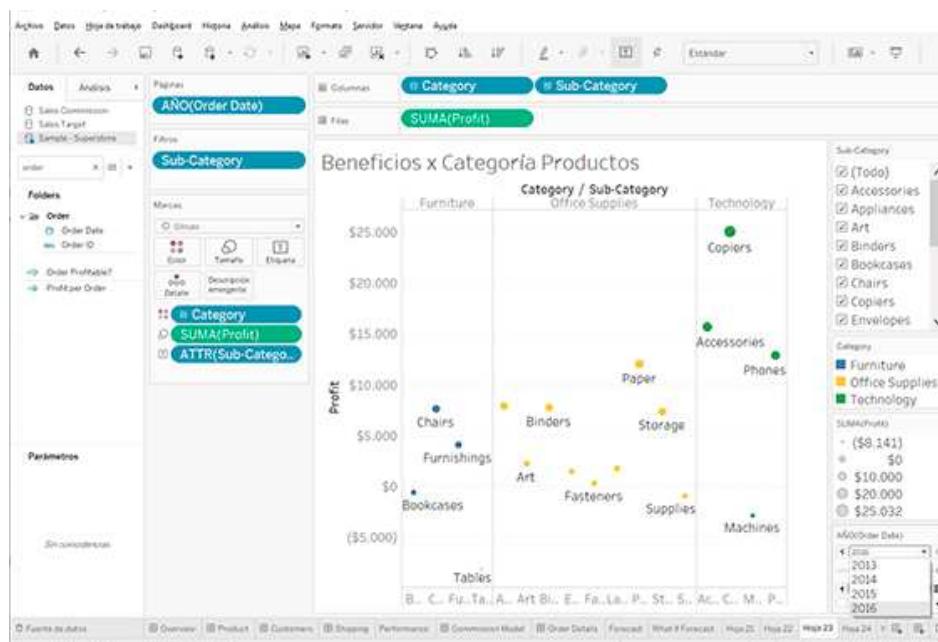
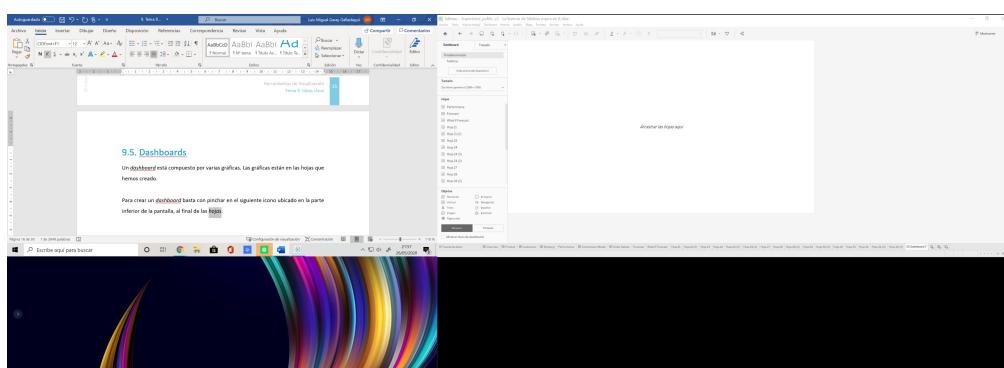


Figura 7. Detalle de las ventas en un año concreto con la opción de Páginas.

10.3. Dashboards

Un *dashboard* está compuesto por varias gráficas o *Dashboards*; las gráficas están en las hojas que hemos creado. Para crear un *dashboard* basta con pinchar en el icono



ubicado en la parte inferior de la pantalla, al final de las hojas.

Pero la verdadera potencia de un *dashboard* no viene solo por la posibilidad de combinar varias gráficas en un mismo espacio, sino porque también podemos vincular los datos entre ellas y establecer relaciones que nos ayuden a analizar los datos de una forma mucho más eficaz. Lo veremos un poco más adelante.

La **composición** de un *dashboard* es muy sencilla: podremos ir construyéndolo solo con arrastrar las hojas situadas en el lateral izquierdo a la zona de trabajo. En este caso, arrastramos dos hojas completas: la que ya conocemos de los Beneficios x Categoría de Productos y una nueva que hemos creado, que representa una distribución de Ventas y beneficios x producto.

El resultado es el siguiente:

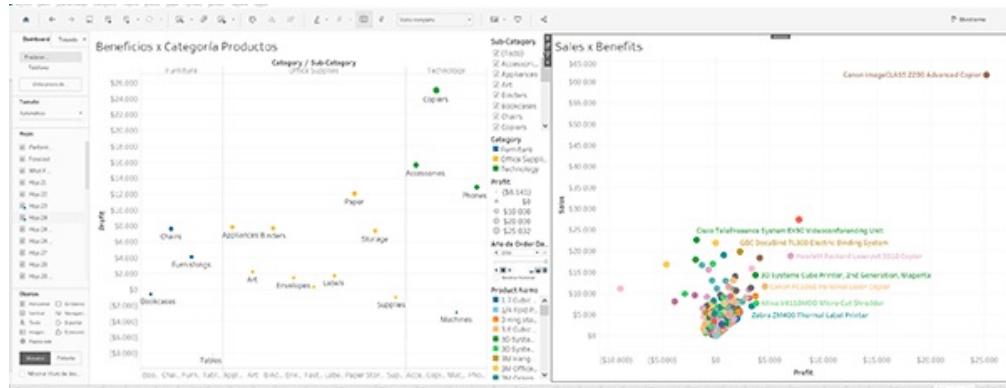


Figura 8. Dashboard con 2 hojas.

Para **vincular los datos** de estas dos hojas que componen el *dashboard* basta con hacer clic en el embudo que aparece en el extremo superior de la gráfica.

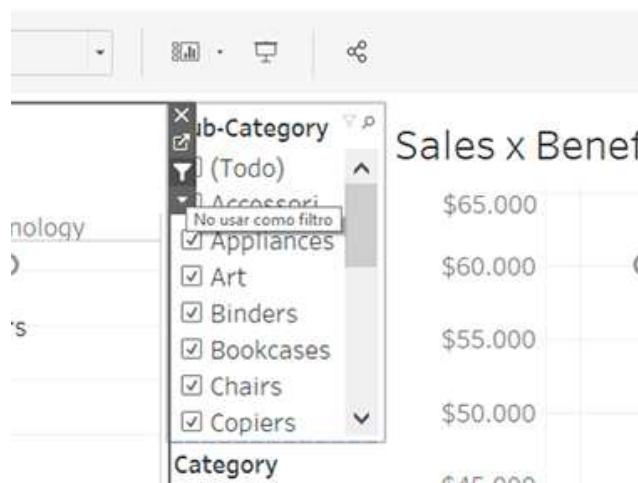


Figura 9. Activación de una hoja como filtro para vincular datos en un *dashboard*.

Y con el filtro ya activado, podemos ver cómo funciona la vinculación de datos entre las hojas, pinchando en cualquier de las subcategorías de la primera hoja. Por ejemplo, al seleccionar en la primera hoja la subcategoría *Paper*, perteneciente a la categoría *Office Supplies*, automáticamente en la segunda hoja del *dashboard* solo

aparecerán los productos de dicha subcategoría.

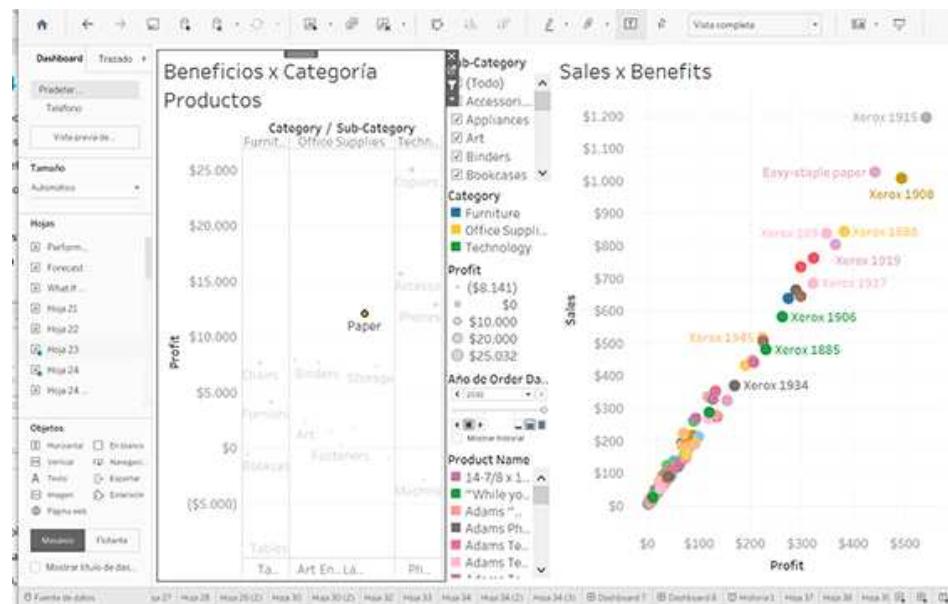


Figura 10. Dos hojas con datos vinculados en un *dashboard*.

Si en el *dashboard* tuviéramos **varias hojas**, también podrías decidir qué hojas vinculamos. En ocasiones puede resultar interesante mantener una hoja fija y el resto vincular sus datos, para ello utilizamos la opción del menú **Dashboard Acciones** y añadimos una opción tipo filtro.

En esta página podemos definir qué hojas de origen y de destino vinculamos. También podemos escoger si, por ejemplo, ejecutamos la acción haciendo clic en la hoja o solamente pasando el ratón por encima de ella.

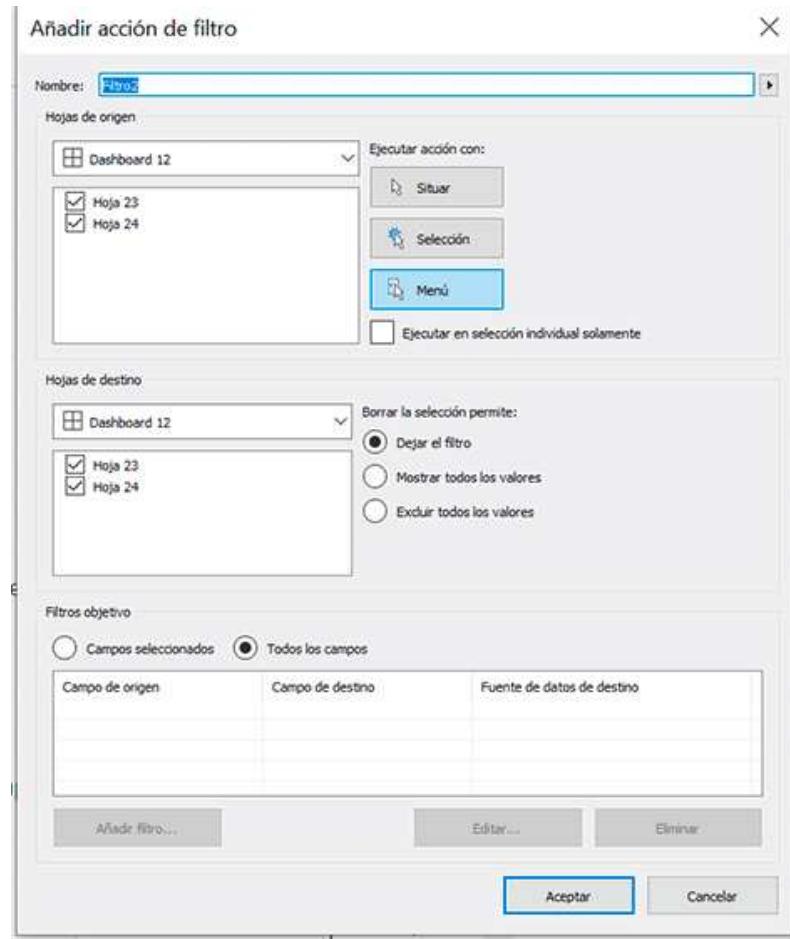


Figura 11. Detalle de añadir un filtro a una hoja de un *dashboard*.

Incluso podemos darle más dinamismo al *dashboard* definiendo que, al hacer clic en una hoja, saltamos a otra diferente del cuaderno, incluso aunque no pertenezca al *dashboard*. Esto nos puede servir para profundizar en el análisis acudiendo a una hoja diferente, que nos puede aportar más detalle, o a una vista complementaria de los datos, etc.

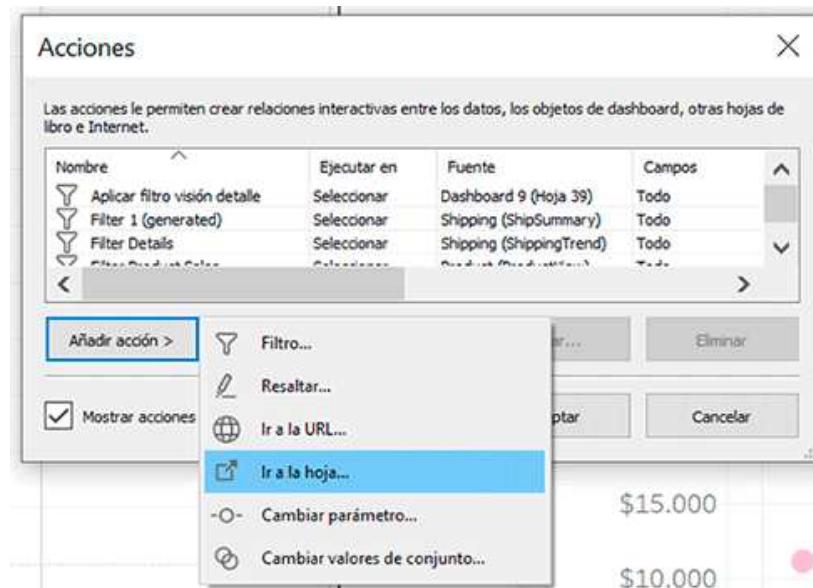


Figura 12. Opciones de Añadir acción en un *dashboard*.

En definitiva, los *dashboard* son un recurso enormemente potente en Tableau que nos permite analizar mejor los datos y nos facilita su comprensión.

10.4. Historias

Una historia nos permite **navegar de forma lineal** entre varias hojas o *dashboards*.

Para crear una Historia, pulsamos en el siguiente botón:



Igual que varias hojas de trabajo forman un *dashboard*, una historia se construye con base en varios *dashboards* u hojas. Lo que se crea realmente en una historia es la **disposición secuencial de hojas y dashboards** bajo el paraguas de la Historia.

Para incluir una hoja o un *dashboard* en una historia, simplemente se arrastran a la parte superior de la Historia. Desde ese momento, ya podemos navegar entre ellas haciendo clic en los botones de navegación de la Historia.

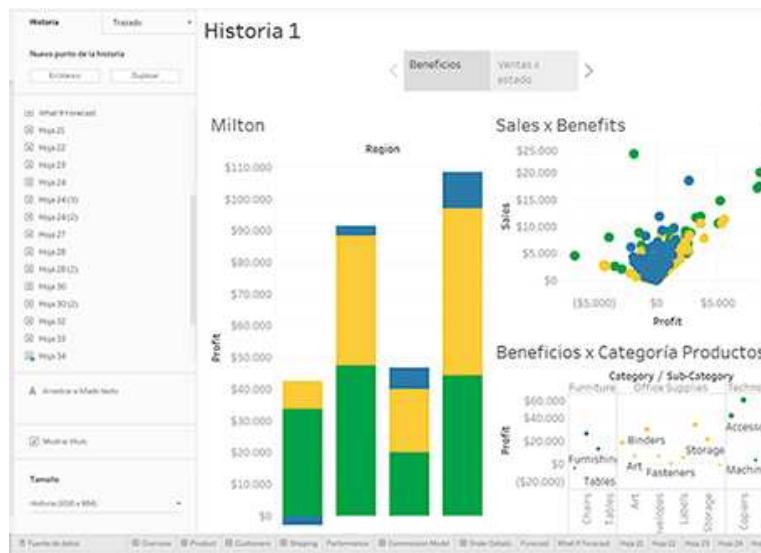


Figura 13. Historia en Tableau.

Una historia es un recurso muy sencillo de utilizar y nos permite construir un auténtico *storytelling* sobre nuestros datos y visualizaciones.

1. Los filtros en Tableau solo permiten emplear los campos asociados a los ejes X e Y de la gráfica:

 - A. Correcto. Si no fuera así la visualización de los datos, se haría imposible.
 - B. Correcto. Solo la versión Desktop permite el uso de filtros.
 - C. Incorrecto. Solo la versión SaaS permite el uso de filtros.
 - D. Incorrecto. Un filtro permite incluir cualquier dato del *dataset*.
2. La definición de un filtro puede incluir todos o parte de los valores de un campo:

 - A. Correcto. Los podemos seleccionar de una lista o incluir todos los valores.
 - B. Correcto. Pulsando F5.
 - C. Incorrecto. Una vez cargados los datos, ya solo nos movemos en su visualización sin filtrarlos.
 - D. Incorrecto. Un filtro por definición contempla todos los valores posibles.
3. La definición de un filtro puede incluir para un campo un valor máximo y mínimo:

 - A. Correcto. Las condiciones de filtro con base en umbrales o condiciones que debe cumplir el filtro.
 - B. Incorrecto. No existe la opción de rangos.
 - C. Incorrecto. Una vez cargados los datos, ya solo nos movemos en su visualización sin filtrarlos.
 - D. Incorrecto. Un filtro por definición contempla todos los valores posibles.

- 4.** La activación de un filtro conlleva su activación automática:

 - A. Correcto. Tras su definición se activa automáticamente.
 - B. Correcto. No es necesario mostrarlo porque ya se activa en el último paso de su definición.
 - C. Incorrecto. Se puede definir un filtro, pero no activarlo.
 - D. Incorrecto. Una vez cargados los datos, ya solo nos movemos en su visualización sin filtrarlos.
- 5.** La tarjeta Páginas nos permite recorrer la visualización y observar los cambios de la misma cuando cambiamos los valores de un campo manteniendo el resto fijos:

 - A. Correcto. Así nos permite cambiar las magnitudes de los ejes de forma dinámica.
 - B. Correcto. Así nos facilita el análisis de los datos, por ejemplo, en su evolución temporal.
 - C. Correcto. Esta opción es incompatible con el uso de filtros.
 - D. Ninguna de las anteriores.
- 6.** Los controles del panel de visualización permiten:

 - A. Hojear las páginas automáticamente.
 - B. Pasar a una página concreta.
 - C. Hacer un recorrido visual de la evolución de los datos de una forma ordenada.
 - D. Todas las respuestas son correctas.

- 7.** Un *dashboard* pueda estar compuesto por:

 - A. Hojas.
 - B. Hojas y *dashboards*.
 - C. *Dashboards*.
 - D. Todas las respuestas son correctas.
- 8.** Un *dashboard* se caracteriza por:

 - A. Incorporar varias gráficas en un mismo espacio de visualización.
 - B. Incorporar indistintamente hojas y otros *dashboards* a su vez.
 - C. La capacidad de vincular los datos de las gráficas que contiene.
 - D. Todas las respuestas son correctas.
- 9.** Una historia nos permite navegar de forma lineal:

 - A. Entre varias hojas.
 - B. Entre varios *dashboards*.
 - C. Arrastrando las hojas y *dashboards* a la parte superior de la Historia.
 - D. Todas las respuestas son correctas.
- 10.** Un *dashboard* nos permite navegar entre diferentes hojas no pertenecientes al cuadro de mando:

 - A. Incorrecto. No se puede salir del contenido del *dashboard*.
 - B. Correcto. Nos permite darle más dinamismo al propio *dashboard*.
 - C. Incorrecto. Solo podemos ir a una página web.
 - D. Incorrecto. Solo podemos movernos entre las hojas del *dashboard*.

Herramientas de Visualización

Tema 11. CARTO. Location Intelligence Software

Índice

[Esquema](#)

[Test](#)

[Ideas clave](#)

[11.1. Introducción y objetivos](#)

[11.2. Inteligencia de ubicación](#)

[11.3. Instalación e interfaz de CARTO](#)

[11.4. Data & Maps](#)

[11.5. Layers & Widgets](#)

[11.6. Estilos](#)

[11.7. Análisis](#)

[A fondo](#)

[QGIS Map Design](#)

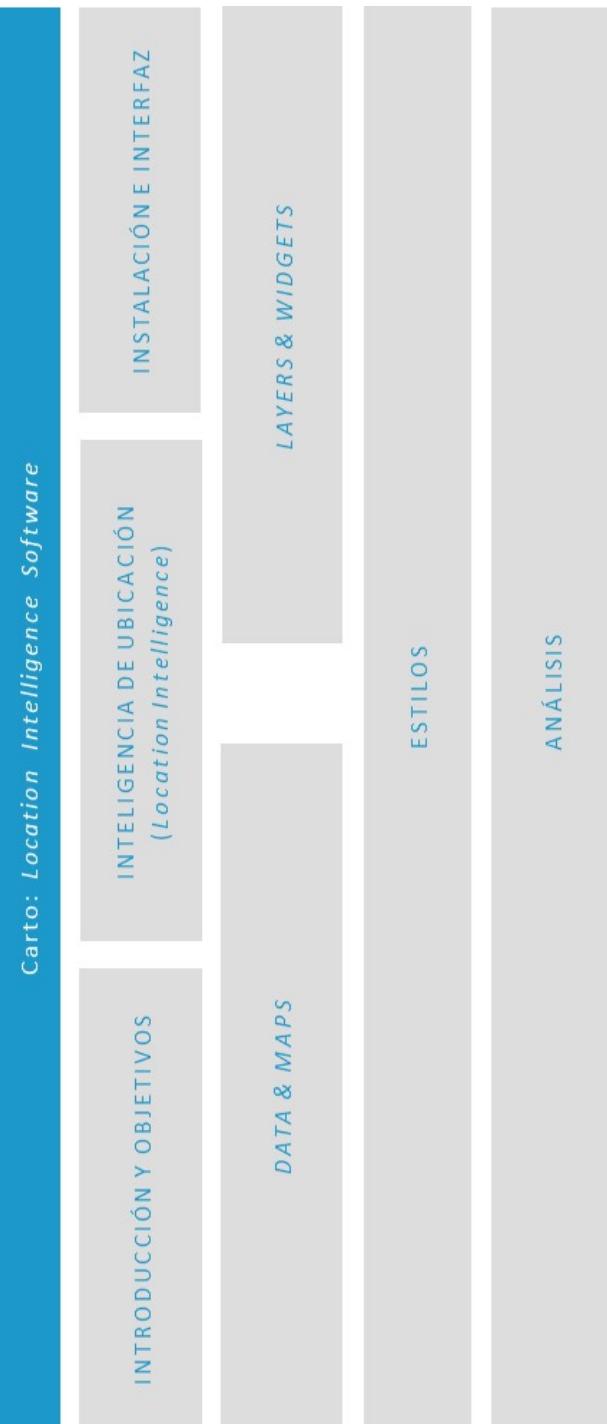
[Introduction to GIS: A free book](#)

[Mapschool](#)

[Tutoriales de aprendizaje de CARTO](#)

[Glosario](#)

Esquema



11.1. Introducción y objetivos

Todo ocurre en algún lugar y, gracias a los sistemas de información geográfica (GIS, del inglés *Geographic Information Systems*), podemos visualizar, analizar e interpretar estos datos espaciales para entender patrones y tendencias.

Las herramientas GIS benefician a las organizaciones de todos los tamaños y en casi todas las industrias. Además, existe un creciente interés por el valor económico y estratégico de los sistemas de información geográfica, siendo CARTO la herramienta de mayor crecimiento del mercado debido al rápido *workflow* desde los datos hasta los *insights*.

Accede a la página oficial de CARTO a través del siguiente enlace:

<https://carto.com/>

CARTO fue fundada en el año 2007 y sus herramientas de geolocalización son utilizadas por compañías como la NASA, Google o el *Wall Street Journal*. CARTO nació como una *startup* tecnológica y sus productos buscan ayudar a grandes empresas a entender mejor sus datos geolocalizados para tomar decisiones de negocio.

Smartphones, sensores, satélites, coches conectados, drones... Los datos de ubicación están cambiando la forma en que vivimos y la forma en que ocurren los negocios. CARTO es una plataforma que convierte los datos de ubicación en rutas de entrega más eficientes, un mejor entendimiento del *marketing* conductual, ubicaciones estratégicas de tiendas y mucho más.

Las personas somos «sensores» en el sentido de que todo lo que hacemos ocurre en algún sitio. Y lo que vamos haciendo se puede quedar grabado como datos. Posteriormente, estos datos se pueden utilizar para entender patrones de comportamiento, de consumo o de cómo la gente actúa en la ciudad.

11.2. Inteligencia de ubicación

El mundo de la inteligencia de ubicación (*Location Intelligence*) cambió cuando los *smartphone* de Apple incorporaron el **sistema de posicionamiento global** (GPS) en 2008. Comenzaron a llegar datos de ubicación de millones de usuarios de teléfonos inteligentes y, ahora, los datos de ubicación están cambiando la forma en que las empresas hacen negocios.

Imaginemos únicamente la cantidad de datos de ubicación que una empresa como Uber procesa cada minuto del día, y Uber no está sola. Las empresas almacenan y procesan los datos de ubicación porque entienden el impacto comercial del «dónde». Sin embargo, el uso de datos de ubicación por parte de la mayoría de las empresas aún está en sus orígenes.

La inteligencia de ubicación (*Location Intelligence*, LI) es una disciplina que pretende convertir los datos de ubicación en resultados comerciales a través del enriquecimiento, la visualización y el análisis iterativo.

LI está dando forma a los negocios del futuro al agregar contexto crítico al proceso de toma de decisiones. Los mapas basados en datos y las aplicaciones de ubicación creadas con LI revelan relaciones y correlaciones espaciales que de otro modo podrían no haber sido tan claras. Mantener una ventaja competitiva en el mercado actual requiere una comprensión total de sus datos de ubicación. Hay más en los datos de lo que parece.

A pesar de que se han logrado avances significativos en el análisis y uso de los datos de ubicación, el objetivo final sigue siendo el mismo: resolver los problemas más desafiantes que enfrentamos hoy.

La utilización moderna de los datos de ubicación comenzó hace más de 30 años para ayudar a los gobiernos a situar en los mapas sus recursos y hacer un mejor uso de ellos. Se mapearon los centros de transporte y las reservas de petróleo, y la industria de sistema de posicionamiento global (GIS) saltó a la fama. Si bien son valiosas, las aplicaciones GIS requieren *software* especializado y capacitación a nivel experto, lo que las hace difíciles de implementar y escalar.

Luego vino Internet, que facilitó un gran avance al hacer que los datos de ubicación estuvieran disponibles en la nube. Los vendedores como Google se convirtieron en pioneros en la democratización del acceso a los datos de ubicación con su producto Google Maps. Esto permitió a los proveedores de inteligencia empresarial (BI) crear ***mashups***, combinaciones de *datasets* discretos que se unieron para crear nuevas aplicaciones. Estos *mashups* superpusieron datos demográficos sobre datos geoespaciales, acercándonos al LI que conocemos hoy.

El panorama actual de la información de ubicación se centra principalmente en analizar y usar los datos de ubicación para obtener información real sobre la sociedad y los servicios. Algunas de las compañías más exitosas de la actualidad, como Uber y

Airbnb, aprovechan los datos de ubicación y colocan los resultados de LI en el centro de sus estrategias comerciales.

LI ayuda a los analistas de negocio y, en general, a cualquier profesional que deba tomar una decisión a obtener información valiosa de una variedad de fuentes de datos de ubicación en crecimiento:

- ▶ ***Big data***: con la creciente disponibilidad de datos de cada máquina, proceso de negocio y transacción de comercio electrónico, las oportunidades de usar datos para resolver problemas son ilimitadas. El análisis de estos datos puede generar ganancias de productividad y la mejora de los procesos comerciales.

- ▶ **Móvil:** los dispositivos móviles brindan información sobre dónde están sus clientes y hacia dónde se dirigen. Esto le permite proporcionar a los clientes información rica en contexto, así mismo, puede entregar alertas e información valiosas y específicas de la ubicación. Los dispositivos móviles también producen información rica en patrones de tráfico, a nivel individual y en conjunto.
- ▶ **Redes sociales:** a partir de estos datos, puede extraer el sentimiento del cliente y aprender lo que los consumidores quieren comprar.
- ▶ **Computación en la nube:** la computación en la nube permite acceder a grandes cantidades de información sin tener que disponer de un *hardware* costoso. Con *Software as a Service* (SaaS), puede comprar servicios que no requieren que se agreguen empleados o personal de atención al cliente. Una empresa puede ampliar sus ofertas según sea necesario sin tener que expandir su negocio.
- ▶ **IoT:** los artículos para el hogar, las máquinas y una serie de otros objetos ahora pueden producir datos utilizables. Por ejemplo, los sensores en los contenedores de basuras pueden ahorrar a una ciudad millones de dólares al comprender cuándo necesitan ser recogidos y optimizar el enrutamiento de recolección de basuras.

11.3. Instalación e interfaz de CARTO

CARTO presenta dos opciones de instalación con el mismo nivel de funcionalidad:

- ▶ *On the cloud*: servicio *online* totalmente gestionado por CARTO en la nube, con *backups* automáticos y actualización continua del *software*.
- ▶ *On-premises*: La plataforma de CARTO se puede desplegar en la propia infraestructura de la empresa, detrás de sus entornos de *firewall* y aplicando las políticas de seguridad en el uso y tratamiento de los datos de la empresa.

En este tema, trabajaremos con la versión *online* que cuenta con herramientas de *drag and drop* para que los analistas y usuarios de negocio puedan descubrir y predecir los principales *insights* de los datos de localización. Permite a los usuarios subir datos y proporciona herramientas para visualizar dichos datos en mapas interactivos.

Para comenzar a usar CARTO, los usuarios primero deben crear una cuenta en línea en (<https://carto.com/signup>). La cuenta básica de CARTO es gratuita y permite a los usuarios cargar una cantidad limitada de archivos.

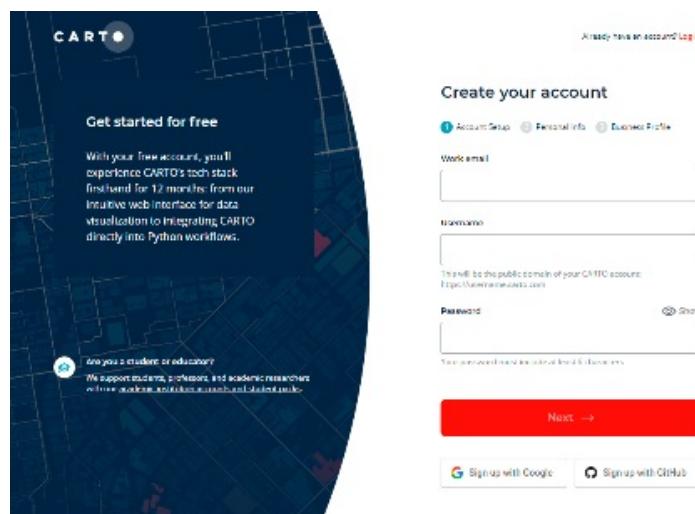


Figura 1. Registro en CARTO.

Una vez creada la cuenta y verificado el *e-mail*, ya podremos empezar a crear nuestro primer mapa.

La página principal dispone de dos grandes apartados:

- ▶ *Maps*: en la vista de **Mapas**, los usuarios pueden construir proyectos interactivos con mapas, combinando uno o más de sus propios conjuntos de datos.
- ▶ *Data*: en la pestaña de **Datos**, los usuarios pueden cargar, editar, ver o eliminar sus propios conjuntos de datos. La forma más sencilla de importar datos a la cuenta de CARTO es arrastrando y soltando el fichero de datos (*drag and drop*) al apartado de datos del *dashboard*.

La **zona de trabajo** tiene una barra de herramientas con opciones para acceder a nuestro *dashboard*, editar el mapa y configurar sus características. Desde aquí podemos gestionar todos los componentes de un mapa: añadir **layers** (capas) y **widgets**, importar *datasets*, personalizar los análisis y cambiar los estilos del mapa.

Iremos viendo cómo se utilizan estas opciones con un caso de uso.

11.4. Data & Maps

Los mapas realizados en CARTO se pueden compartir y embeber en sitios web y en diferentes formatos. Una de las ventajas de utilizar CARTO en comparación con otras plataformas de mapeo interactivo es que nos permite cargar directamente los conjuntos de datos y manipularlos en línea. Esta funcionalidad incluye seleccionar partes específicas del conjunto de datos utilizando consultas de estilo SQL.

Comenzaremos a utilizar CARTO cargando el conjunto de datos «carto-tutorial.data.csv».

Accede al conjunto de datos «carto-tutorial.data.csv» a través del siguiente enlace:

<https://github.com/alonsocap/CARTO-tutorial/tree/master/data>

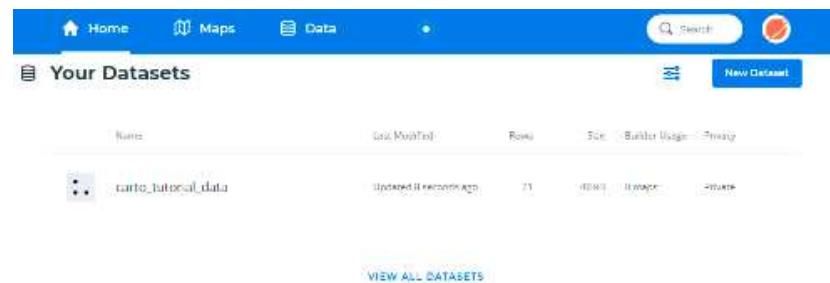


Figura 2. *Datasets* disponibles en CARTO.

Los datos que usaremos estarán en un **formato tabular**, representando puntos de ubicación para tres personas a medida que viajan desde diferentes lugares de partida, y terminan en París, Francia. Veamos de un vistazo a la estructura de datos.

Los datos están formateados con 7 columnas:

	A	B	C	D	E	F	G	H
1	person	sequence	lat	lon	location	country	id	
2	James Howlett	1	46.856255	-73.255688	Quebec	Canada	1	
3	James Howlett	2	46.8214974	-71.2537732	Quebec City	Canada	1	
4	James Howlett	3	43.6800052	-70.331302	Portland	Maine	1	
5	James Howlett	4	47.4823015	-53.1095691	St. John	Nova Scotia	1	
6	James Howlett	5	46.1621532	-1.2464739	La Rochelle	France	1	
7	James Howlett	6	47.3943622	0.6247271	Tours	France	1	
8	James Howlett	7	48.8589996	2.2071248	Paris	France	1	
9	Natasha Romanoff	1	58.304648	83.847923	NA	Russia	2	
10	Natasha Romanoff	2	46.127785	64.072527	NA	Kazakhstan	2	
11	Natasha Romanoff	3	40.90655	33.047131	NA	Turkey	2	
12	Natasha Romanoff	4	49.718523	23.862555	NA	Ukraine	2	
13	Natasha Romanoff	5	50.0593307	14.1847427	NA	Czech Rep.	2	
14	Natasha Romanoff	6	50.1974403	9.9145805	Bad Kissigen	Germany	2	
15	Natasha Romanoff	7	48.8587737	2.2071267	Paris	France	2	
16	Ami Han	1	37.5647673	126.7086576	Seoul	South Korea	3	
17	Ami Han	2	31.2231275	120.9148396	Shanghai	China	3	

Figura 3. Detalle de «carto-tutorial-data.csv».

Cada una de estas tres personas ha registrado su ubicación a lo largo de 7 paradas, ya que hacen su camino alrededor del mundo hasta París. **Cada fila representa una parada**. En este caso nos interesa **mapear el movimiento de estos individuos**, siguiendo su secuencia de paradas. No tenemos información sobre cuánto tiempo pasan en cada parada, pero sí sabemos el orden en que viajan según la columna de secuencia. También sabemos qué persona se detiene porque tenemos una columna de identificación y una columna con su nombre.

Ahora que tenemos una idea de cómo son nuestros datos y que ya hemos subido el fichero a CARTO, podemos comenzar a explorarlo y visualizarlo.

CARTO almacena los datos como si fueran una tabla en una base de datos PostgreSQL. A todo *dataset*, CARTO le añade 2 campos principales visibles:

- ▶ `cartodb_id`: clave principal del *dataset*, tiene un valor único y no puede ser null.
- ▶ `the_geom`: campo donde se mapean los puntos geométricos y que en nuestro caso se basa en las columnas lat y lon.

id	name	neighborhood	lat	lon	state	country	id
1	Brooklyn Heights	Brooklyn	40.6892	-74.0445	New York	United States	1
2	Dumbo	Brooklyn	40.6992	-74.0145	New York	United States	2
3	Greenpoint	Brooklyn	40.7092	-74.0105	New York	United States	3
4	Williamsburg	Brooklyn	40.7192	-74.0045	New York	United States	4
5	Brooklyn Bridge Park	Brooklyn	40.7292	-74.0005	New York	United States	5
6	Sloans Hill	Bronx	40.8192	-73.9845	New York	United States	6
7	Grand Concourse	Bronx	40.8292	-73.9805	New York	United States	7
8	East Tremont	Bronx	40.8392	-73.9765	New York	United States	8
9	South Bronx	Bronx	40.8492	-73.9725	New York	United States	9
10	West Bronx	Bronx	40.8592	-73.9685	New York	United States	10
11	Bruckner	Bronx	40.8692	-73.9645	New York	United States	11
12	University Heights	Bronx	40.8792	-73.9605	New York	United States	12
13	Washington Heights	Bronx	40.8892	-73.9565	New York	United States	13
14	Harlem	Bronx	40.8992	-73.9525	New York	United States	14
15	Manhattan Bridge Park	Bronx	40.9092	-73.9485	New York	United States	15

Figura 4. Detalle de los datos.

Toda esta información la podemos mostrar en un mapa. En la parte inferior de la tabla existe un botón que nos permite previsualizar estos datos o ya directamente **crear un mapa**.

Recuerda que cada fila de nuestra tabla suponía una parada. Luego en la vista de mapa, los puntos reflejan las paradas de nuestros viajeros. Cuando los puntos se visualizan por primera vez, todos se representan con el mismo color y no se aprecia ninguna diferencia.

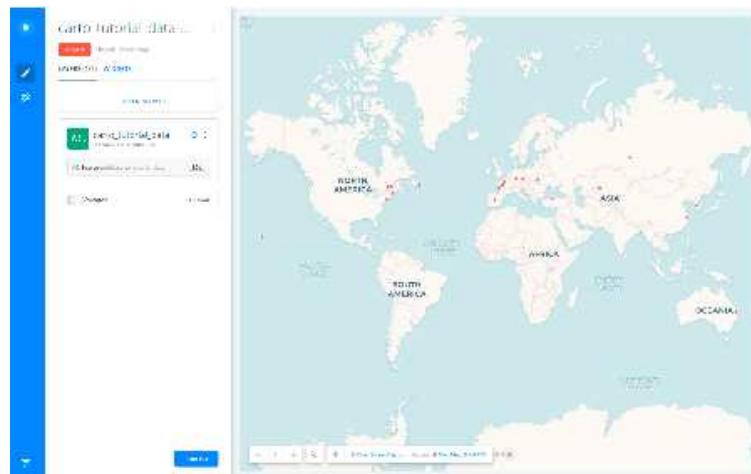


Figura 5. Ubicación de las paradas de nuestros viajeros.

Para **modificar la apariencia** del mapa, podemos empezar por cambiar el fondo.

Para ello disponemos de la opción **BaseMap** donde escogeremos entre diferentes fuentes y estilos.

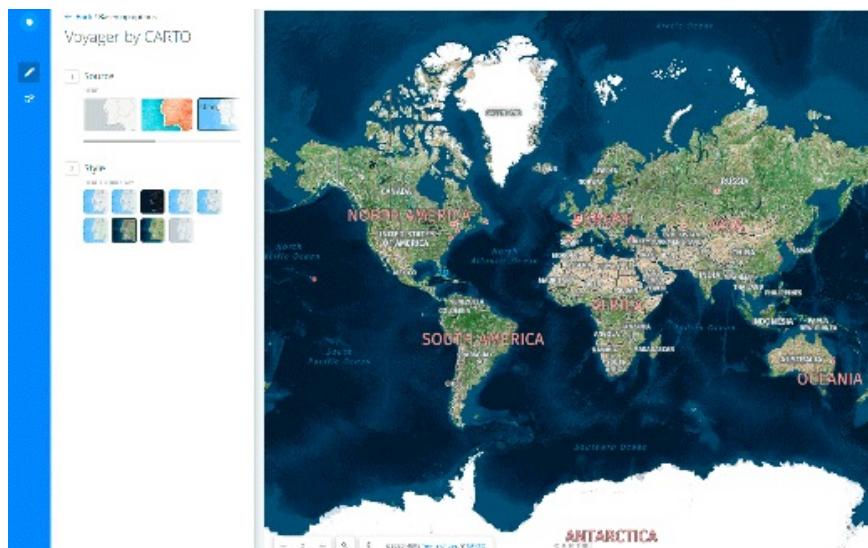


Figura 6. Opciones de BaseMap.

11.5. Layers & Widgets

CARTO puede manejar diferentes capas o *layers*, lo que permite **solapar sobre un mismo mapa diferentes fuentes de información**. Posteriormente se pueden habilitar o deshabilitar la visualización de dichas capas, permitiendo analizar los datos de forma flexible.

Para habilitar el selector de capas basta con marcarlo desde la opción **Map Options** ubicada en el lateral izquierdo. En nuestro caso solo disponemos de una capa, pero se puede apreciar cómo podemos habilitarla y deshabilitarla de una forma muy sencilla.



Figura 7. Selector de capas o *layers*.

Los *widgets* están embebidos en nuestra visualización, aunque no modifican los datos. Simplemente nos permiten explorar el mapa seleccionado diferentes tipos de filtros.

Existen diferentes tipos de *widgets*:

- ▶ **Categoría:** permite agregar datos con diferentes métodos, y crear categorías.
- ▶ **Histograma:** examina los datos numéricos dentro de un rango determinado.
- ▶ **Fórmula:** calcula valores agregados a partir de columnas de tipo numérica, como AVG (media), MAX (máximo), MIN (mínimo), SUM (suma) y COUNT (contar).
- ▶ **Serie de tiempo:** hace posible que se muestre la información de forma animada (agregando los datos) a lo largo del tiempo.

Para **añadir widgets** lo podemos hacer a través de la opción Widgets ubicada junto a la opción Layers. Una vez seleccionados aparecerán en la parte derecha del mapa y nos sirven para filtrar la información.

Todos los datos de los filtros están vinculados entre sí, de tal manera que, en nuestro caso, si por ejemplo seleccionamos un país como China, veremos las localizaciones que han atravesado nuestros viajeros. En concreto solo Ami Han viajó a través de China y concretamente paró en Shanghai.

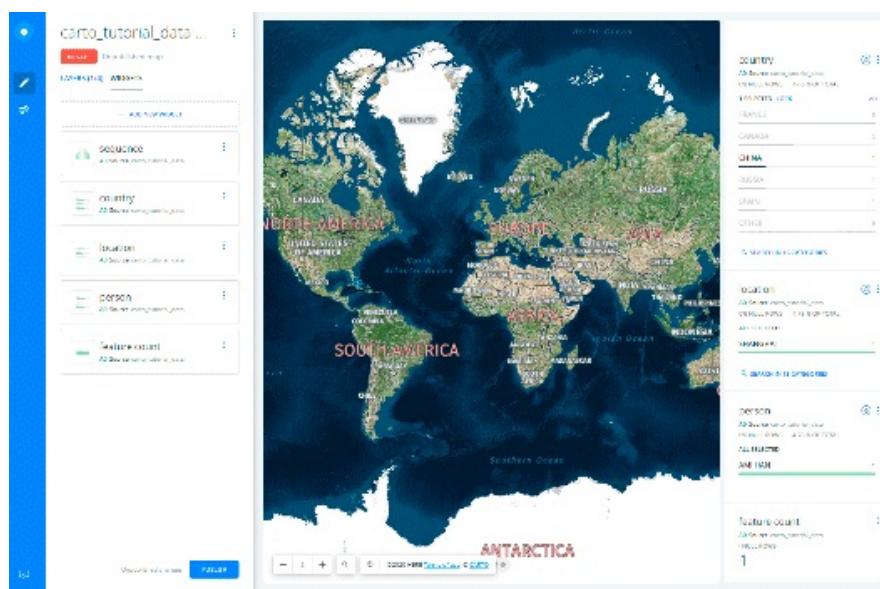


Figura 8. Selector de *widgets*.

A través de los filtros que tenemos creados también podemos fácilmente, por ejemplo, identificar las ciudades que han atravesado cada uno de nuestros 3 viajeros. En concreto, seleccionamos el caso de Natasha Romanoff y vemos que se ha movido por Europa y Asia.



Figura 9. Uso del *widget persona*.

11.6. Estilos

CARTO permite trabajar con los estilos (*Style*). Para ello pinchamos directamente en el *layer* que queramos configurar su estilo y seleccionamos la opción **Style**.

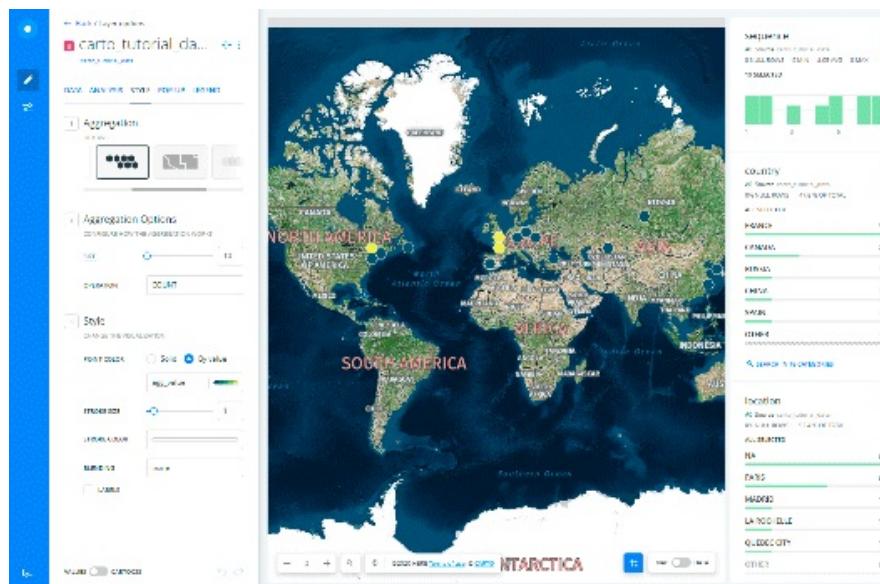


Figura 10. Estilos de una *layer*.

En los estilos hay dos bloques principales de opciones:

- ▶ **Agregación:** permite emplear diferentes formas de agregar los datos y mostrarlos.
En nuestro caso seleccionamos el estilo de hexágonos y agregamos los datos con la operación count porque queremos mostrarlos todos.
- ▶ **Style:** dispone de opciones para cambiar el tamaño de los puntos, su color y resolución. También nos permite trabajar con la forma de las etiquetas.

11.7. Análisis

CARTO ofrece una inmensa oferta de análisis sobre los datos cartográficos, permitiendo vincular datos entre capas, crear agrupaciones naturales de puntos en función de la proximidad o predecir tendencias o crear áreas de influencia. Algunos de estos análisis pueden ser de pago y estar solo disponibles en la versión Enterprise.

Veamos algunos casos de uso que se encuentran distribuidos en los 3 siguientes bloques:

- ▶ *Create and clean*: ofrece la posibilidad de **añadir datos** económicos, demográficos o sociales a nuestros datos, para realizar análisis más complejos. También se podrían **filtrar** los datos de una capa a partir de una segunda capa o unir información procedente de diferentes conjuntos de datos.
- ▶ *Analyze and predict*: permite realizar **agrupaciones de datos** en función de su proximidad geográfica o de otros criterios, identificando también aquellos puntos denominados *outliers* que no cumplen el patrón con el que se agrupan los datos.
- ▶ *Transform*: permite realizar la **intersección entre diferentes capas**, crear áreas de influencia, calcular el centroide para todo un conjunto de datos o agrupar los puntos en polígonos. También podemos crear capas de líneas conectando puntos entre sí.

En nuestro caso, vamos a unir los diferentes puntos en los que ha parado cada viajero en el orden en que lo han hecho, modificando además el color de cada ruta para distinguirlos.

Para ello seleccionamos el tipo de análisis *Create Lines from Points* y escogemos que sea de tipo secuencial para unir todas las paradas de cada viajero. Además, para distinguir las paradas de cada uno de ellos y que no se mezclen los trayectos, seleccionamos como orden de agrupación de los puntos el campo person y así lo configuramos en la opción Group By.



Figura 11. Análisis tipo *Create Lines from Points*.

Adicionalmente, para poder distinguir mejor los diferentes itinerarios que hemos creado, nos puede interesar marcarlos con diferentes colores. Para ello vamos a la opción Style > Lines style, seleccionamos dentro de Stroke color el campo category que contiene el nombre de cada viajero.



Figura 12. Estilo de la línea de itinerarios.

El resultado final es el que se muestra a continuación:



Figura 13. Rutas marcadas en diferentes colores.

En la visualización se pueden observar 3 rutas diferentes: la ruta roja es la de James Howlett, la verde es de Ami Han, quien ha visitado los lugares más alejados entre sí, y Natasha Romanoff tiene la ruta blanca.

Y como decíamos al principio, todas las rutas confluyen en París.

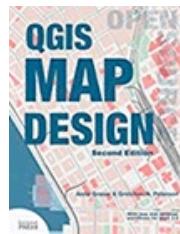


Figura 14. Viajando por el mundo y llegando a París.

Las opciones que ofrece CARTO para el tratamiento de datos geoespaciales son enormes. Al ser una herramienta especializada para ello, supera la simple ubicación de los datos en un mapa que proporcionan otras herramientas y ofrece posibilidades que merecen la pena explorar y disfrutar como si fuéramos auténticos mercaderes y viajeros al estilo Marco Polo.

QGIS Map Design

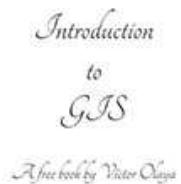
Graser, A., Peterson, G. N. y Sherman, G. (2018). *QGIS Map Design*. Locate Press.



Este libro desarrolla los principios de QGIS a través de las últimas técnicas en el uso de herramientas para acelerar el diseño de diseño de mapas, incorporando las tendencias y tecnologías más novedosas.

Introduction to GIS: A free book

Graser, A., Peterson, G. N. y Sherman, G. (2018). *QGIS Map Design*. Locate Press.
Recuperado de <https://volaya.github.io/gis-book/en/index.html>



Este libro gratuito es una completa introducción a GIS que te ayudará a entender los sistemas de información geográfica.

Mapschool

Mapschool. Página web oficial. <https://mapschool.io/>

Se trata de una introducción rápida y completa sobre diferentes conceptos relacionados con la cartografía.



Tutoriales de aprendizaje de CARTO

Carto. (S. f.). *Tutorials: Step-by-step guides and instructions to help you get started and get to the next level.* <https://carto.com/help/tutorials/your-account/>

En esta página web encontrarás guías de referencia rápida para aprender las características individuales de CARTO.



Glosario

Carto. (S. f.). Glossary: Find CARTO-related terminology.

<https://carto.com/help/glossary/>

Este glosario de CARTO contiene definiciones muy útiles sobre la mayoría de los conceptos, componentes y tecnologías relativas a GIS.



1. ¿Cuál fue el primer fabricante de móviles que incluyó GPS?

 - A. LG.
 - B. Apple.
 - C. Samsung.
 - D. Xiaomi.

2. ¿En qué año salió a la venta el primer móvil con GPS?

 - A. 2000.
 - B. 2006.
 - C. 2008.
 - D. 2011.

3. En la pestaña *Data* de CARTO, los usuarios pueden:

 - A. Cargar, editar, ver o eliminar datos.
 - B. Elegir entre conjuntos de datos proporcionados por la aplicación.
 - C. Escoger el tipo de visualización de sus datos.
 - D. Añadir *widgets* al mapa.

4. Las relaciones y correlaciones espaciales entre los datos no se pueden descubrir en:

 - A. Los mapas basados en datos.
 - B. Las aplicaciones de ubicación creadas con LI.
 - C. Las visualizaciones clásicas como diagramas de barras o tartas.
 - D. Los mapas con geoposicionamiento.

5. Internet supuso un gran cambio en LI al hacer que:
 - A. Todo el mundo pudiera consultar Google Maps.
 - B. Se usarán aplicaciones de mensajería gratuitas.
 - C. Los datos de ubicación estuvieran disponibles en la nube.
 - D. Todas las anteriores.

6. CARTO es una plataforma en línea:
 - A. Verdadero. Dispone de la opción de uso en la nube.
 - B. Verdadero. Es la única opción.
 - C. Falso. No dispone de la opción en la nube.
 - D. Falso. El rendimiento de los mapas es muy exigente para emplear una versión en la nube.

7. La inteligencia de ubicación (LI) es una disciplina que pretende:
 - A. Convertir los datos de ubicación en resultados comerciales.
 - B. Mejorar la información de los mapas y GIS.
 - C. Vender los datos geográficos a las empresas.
 - D. Masificar el conocimiento de los mapas y GIS.

8. CARTO no permite seleccionar ciertas partes específicas del conjunto de datos ni utilizar consultas SQL:
 - A. Verdadero. El mapa refleja toda la información del *dataset*.
 - B. Falso. Se puede escoger la información del *dataset* a visualizar en el mapa.
 - C. Verdadero. CARTO no admite SQL.
 - D. Falso. Puede seleccionar información parcial del dataset pero no admite consultas SQL.

9. En la Vista de mapa, los datos se visualizan como puntos y cada punto representa una fila de datos:
- A. Falso. Cada punto representa un valor de cada columna.
 - B. Verdadero. Cada punto representa una fila.
 - C. Falso. Los datos no reflejan las filas del *dataset*.
 - D. Falso. Los puntos no reflejan las filas del *dataset*.
10. Las dimensiones son los datos descriptivos, mientras que las medidas son:
- A. Datos ordinales.
 - B. Datos numéricos.
 - C. Las dimensiones del contenedor gráfico.
 - D. Las dimensiones de los datos descriptivos. Son equivalentes.