Cápsula 2: Agregación e histogramas con D3.js

Hola, bienvenidxs a una cápsula del curso Visualización de Información. En esta revisaremos algunas funciones de agregación de D3.js, y las aplicaremos para generar vistas de histogramas.

Anteriormente mencionamos algunas funciones de utilidad que D3.js provee para realizar cálculos agregados sobre los datos a mostrar. Cómo “d3.min” y “d3.max”, que usamos para definir escalas y así repartir el espacio de nuestras visualizaciones.

El sub paquete “d3-array” contiene varias de estas funciones, que permiten cálculos agregados simples, y hacen uso de funciones de acceso para los atributos a utilizar para los cálculos. La función “d3.mean” por ejemplo permite calcular el promedio de los valores accedidos.

Este programa de prueba, que se muestra en pantalla, carga el clásico *dataset* de Iris, y luego calcula promedios para cada atributo cuantitativo del *dataset*. El realizar este tipo de cálculos en el mismo programa que genera las visualizaciones es una decisión de implementación a tomar. Si en este caso lo único que utilizaremos son los promedios, entonces tal vez conviene preprocesar esos valores, así se ahorra esa computación el programa de visualización.

Pero no siempre es el caso donde conviene. Puede ser que en un caso específico se necesite de todos los datos desagregados y de información agregada. Un caso son las particiones de datos, donde podemos hacer uso de tanto los elementos individuales de un *dataset*, como de agrupaciones del mismo.

El paquete “d3-array” provee también funciones para generar particiones de nuestros datos. Una básica es “d3.group” que genera grupos de un iterable en base a quienes comparten el valor de algún aspecto, como un atributo. Por ejemplo, el código en pantalla genera grupos de los datos de Iris según el valor del atributo de especie.

El método “d3.group” por defecto retorna un objeto de tipo “Map”, que es un tipo de diccionario que existe en JavaScript. Aquí se lo entregamos a “Object.fromEntries” para que lo transforme al tipo objeto, que es lo que solemos usar. Si lo probamos, vemos un objeto con tres llaves: las tres especies de Iris en el *dataset*, y cada llave asociada a un arreglo con 50 objetos. Como podrás imaginar, cada uno de estos elementos comparte el valor de especie del grupo.

Además de “d3.group”, hay funciones de utilidad para casos más complicados. Por ejemplo, “d3.rollup” permite crear una partición y agrupar de forma similar a “d3.group”, pero además calcula un resultado a partir de cada grupo.

El ejemplo en pantalla, genera grupos en base al valor del atributo especie, y luego por cada grupo retorna el largo del grupo. Entonces esto nos entregaría un contador de ítems por grupo. Otro ejemplo es el que sigue, que genera grupos de la misma forma, pero entrega un objeto por cada grupo que promedia todos los atributos cuantitativos. De esa forma, obtenemos un elemento representante por cada grupo.

Si los probamos, vemos que de forma similar a “d3.group”, al pasar por “Object.fromEntries” nos retorna un objeto con las llaves correspondientes a los valores usados para agrupar. En cada caso, asociado al resultado de “d3.rollup”: un contador y en objetos con promedios.

El tipo de particiones revisado hasta el momento suele generarse con atributos categóricos u ordinales, mientras que para atributos cuantitativos se suelen agrupar en rangos de valores. La función “d3.bin” permite realizar esto mismo, y generar grupos de objetos por rangos de valores numéricos.

A diferencia de las funciones anteriores, llamar a “d3.bin” entrega un objeto generador que se puede ocupar como función para generar grupos en base a datos. En su forma más básica, esperaría recibir un arreglo de números para repartir. Si le entregamos un arreglo de objetos, entonces mediante el método “value” se puede especificar la función de acceso al valor que usar para separar.

En el ejemplo en pantalla se eligió utilizar el largo de sépalo de las flores, y luego se llama la función separadora sobre nuestro arreglo de datos. Si lo probamos y revisamos el resultado, notaremos que nos entrega un arreglo de ocho arreglos, cada uno es un rango distinto. Si vemos los detalles, vemos que cada rango tiene un grupo de datos individuales, además de atributos “x0” y “x1” que especifican los valores extremos de ese rango.

Por defecto, el número de rangos resultante se determina a partir de los datos entregados, y en este caso entregó ocho. Se puede alterar cómo se realiza esa división por rangos llamando a otros métodos del objeto entregado por “d3.bin”. ¡Investigalos buscando en la documentación! Por ejemplo, si quisiera dividir todo en solo dos rangos, ¿cómo lo haría con “d3.bin”? ¡Déjalo en los comentarios!

Este método es particularmente útil para generar histogramas de atributos. Como sabemos, este *idiom* suele codificar información previamente agregada, y al ocuparse con atributos cuantitativos, se suele ordenar por rangos. El resultado que retorna la función nos permitiría crear barras cuyo largo codifican el largo de cada grupo en un rango.

Con eso en mente, extendí nuestro ejemplo típico de gráfico de dispersión y agregué vistas yuxtapuestas que muestran la distribución de los atributos individuales. Para esta codificación fue necesario manejar de forma independiente los espacios para cada *idiom*: uno para el gráfico de dispersión, y dos para histogramas.

El gráfico de dispersión codifica directamente elementos individuales como puntos. Mientras que los histogramas se generaron a partir de agregaciones usando “d3.bin”: una

que agrupa en base al atributo codificado en el eje X, y otra que agrupa en el atributo en el eje Y.

El resto es bastante similar a lo que ya hemos revisado. Algo que fue distinto es que se usó el método “domain” de “d3.bin”. Averigua por tu cuenta qué efecto tiene esto, e intenta incluso sacar esa sentencia del código y ve que ocurre gráficamente. Si crees que lo entendiste, ¡déjalo en los comentarios del video!

Con eso termina el contenido de esta cápsula. Recuerda que si tienes preguntas, puedes dejarlas en los comentarios del video para responderlas en la sesión en vivo de esta temática. ¡Chao!