Cápsula 3: Gráfico de dispersión agregado y hexbin

Hola, bienvenidxs a una cápsula del curso Visualización de Información. En esta revisaremos otro ejemplo práctico usando D3.js de agregación, específicamente utilizando hex-bin y generaremos un gráfico de dispersión agregado.

En pantalla muestro nuevamente una adaptación de nuestro ejemplo recurrente de gráfico de dispersión. Esta vez lo adapté para que se generarán 5000 elementos de forma aleatoria siguiendo distribuciones de probabilidad en cada atributo.

Con esto podemos apreciar de forma gráfica el problema de oclusión que puede parecer en gráficos de dispersión. Al contar con 5000 elementos que se concentran en algunas regiones, podemos apreciar la concentración, pero no la cantidad relativa entre secciones.

Como propuesta de un *idiom* que si arregla esto propusimos el gráfico de dispersión continuo, que codifica pixel por pixel la concentración de elementos. Ahora haremos una adaptación de esa idea, pero realizaremos un punto medio, donde agregaremos elementos por regiones espaciales más grandes que pixeles.

D3 provee una herramienta para realizar agregación en un espacio bidimensional en regiones hexagonales pequeñas: "d3-hexbin". Al llamar a la función "d3.hexbin", se obtiene un generador de partición espacial, similar a "d3.bin" que vimos en la cápsula pasada.

Pero a diferencia de este último, "hexbin" se espera realizar sobre coordenadas del monitor, o en nuestro caso, las coordenadas utilizadas dentro del SVG para ubicar elementos. El generador "bin" regular era en base a valores de datos, aquí nos enfocamos en las coordenadas donde hubiéramos mostrado elementos.

Con eso en mente, el generador de particiones tiene métodos que ayudan a realizar particiones espaciales en pequeños hexágonos. El ejemplo en pantalla muestra algunos que usamos para nuestro caso.

El método "extent" permite definir la región espacial a dividir según coordenadas relativas, y el método "radius" especifica el radio de los hexágonos a generar. Aquí entregamos el tamaño de la región que contiene puntos en el gráfico original, y un radio arbitrario de 7.

Luego, los métodos "x" e "y" son similares a otros objetos generadores que hemos visto. Estos permiten determinar cómo calcular las coordenadas espaciales de cada dato, que se usarán para realizar la partición espacial general. Así podemos entregarle al generador datos crudos y no debemos realizar un procesamiento que calcule de forma separada cada coordenada.

Podemos probar un llamado del generador sobre los datos del ejemplo. Similar a las funciones de agregación revisadas, obtenemos un arreglo grande de arreglos con los grupos de puntos. Cada grupo contiene los objetos que son vecinos espacialmente, y tiene atributos “x” e “y” que hacen referencia al centro del hexágono correspondiente al grupo.

¡Nos falta ver los hexágonos aún! Para eso, agregaré un elemento SVG completo aparte, y aplicaré los mismos ejes que el gráfico de dispersión original, de forma que veamos ambos *idioms* lado a lado.

También, replicaré la idea de usar color para codificar la concentración de elementos en cada grupo. Por eso creo una escala que representa un *colormap* secuencial en base a la cantidad de elementos por grupo.

Y por último falta agregar las figuras hexagonales visuales. De forma similar a los símbolos complejos que revisamos, podemos crear hexágonos mediante elementos “path” que tengan una forma de hexágono. Para eso, “hexbin” provee un método que retorna la forma de hexágono centrado en el origen. Con las coordenadas de hexágonos por grupo podemos trasladar cada uno, y con el *colormap* podemos rellenar su color.

Si probamos el código, ¡vemos que funciona! Podemos apreciar comparando lado a lado, que solo se generan hexágonos en las secciones espaciales donde hay puntos en el original. También podemos apreciar de mucha mejor forma dónde hay concentraciones importantes de puntos, lo cual antes es casi invisible, por culpa de la oclusión de puntos.

Como los otros ejemplos de este grupo, tendrás acceso a este código. Te invito a revisarlo, jugar alterando parámetros y ver los resultados que dan. En la documentación de “d3-hexbin” podrás encontrar ejemplos similares e incluso otros que toman otras direcciones. Como en vez de codificar concentración mediante color, codificar mediante tamaño de hexágonos.

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!