Uso de *layouts*.

Los *layouts* en d3.js son objetos que reciben un conjunto de datos con el objeto de manipularlo, transformarlo y generar nueva información de manera conveniente para una visualización específica posterior. Existe una amplia lista de *layouts* relacionados con distintos tipos de gráficos: Pie, Histogram, Pack, Chord, Treemap, Force...

En esta primera parte del boletín, exploraremos únicamente el funcionamiento de Pack y Chord, pero se anima al alumno a analizar la gran variedad de ejemplos de la galería oficial:

https://github.com/mbostock/d3/wiki/Gallery

En primer lugar, trabajaremos sobre el ejemplo *Circle Packing* de la galería oficial de d3.js. Se pide:

- Descargar el código del ejemplo (junto con los datos) y ponerlo en marcha en el equipo local.
- Analizar el funcionamiento del *layaout* de tipo Pack. ¿Dónde y cómo recibe los datos del fichero en formato JSON?
- Invocar a la función padding del layout para modificar la distancia entre los círculos (https://github.com/mbostock/d3/wiki/Pack-Layout).

Ahora, pasemos a utilizar ejemplo *Zoomable Pack Layout* de la galería oficial de d3.js. Se pide:

- Descargar el código del ejemplo (junto con los datos) y ponerlo en marcha en el equipo local. Comentar la importación del fichero d3.layout.pack.js, ya que no es necesario con la nueva versión de la librería.
- Analizar las diferencias en el uso del *layaout* de tipo Pack. ¿Qué papel juega el método value(...)? ¿Cómo recibe los datos del fichero en este caso?
- Modifica la función zoom(...) para que la transición sea más lenta si se pulsa la tecla Shift.
- Modifica el fichero de datos para que contenga una jerarquía con el número de alumnos matriculados en las titulaciones de la Universitat de València. La estructura de dicha jerarquía será: UV → Facultad → Departamento → Grado (nombre y número de alumnos).

El *layout* Pack también se puede usar para crear gráficos de burbuja como los que muestra el ejemplo *Bubble Chart* de la galería oficial de d3.js. Para este ejemplo se pide:

- Descargar el código del ejemplo (junto con los datos) y ponerlo en marcha en el equipo local.
- Modificar la ordenación a d3.ascencing o d3.descending y comprobar el funcionamiento.

- Substituir el fichero de datos por el que se ha generado en el paso anterior sobre alumnos matriculados en los grados de la Universitat de València.

- Intercambiar la escala de colores por una de tipo d3.scale.category10.

Los diagramas de tipo Chord muestran las relaciones 1:1 existentes entre un grupo de entidades, representadas a través de una matriz cuadrada. El ejemplo *Euro Debt* de la galería oficial de d3.js muestra una aplicación de rabiosa actualidad de la que se pueden extraer conclusiones interesantes.

En una versión más sencilla, el ejemplo *Chord Diagram* de la galería oficial d3.js realiza las mismas tareas. Se pide:

- Descargar el código del ejemplo (junto con los datos) y ponerlo en marcha en el equipo local.

- Modifica la matriz de datos para que se relacionen 5 valores y añade un nuevo color a la escala ordinal de la variable “fill”.

- Modifica el espaciado y la ordenación de los grupos, los subgrupos y los chords. Las funciones necesarias para ello están explicadas en el API: https://github.com/mbostock/d3/wiki/Chord-Layout

Exportación de gráficos SVG.

La exportación de contenido gráfico a un fichero se puede hacer de forma directa gracias a d3.js. El ejemplo *Word Count* de la galería oficial de d3.js muestra cómo se puede generar una imagen de tipo raster (PNG) o de tipo vectorial (SVG). Se pide

- Analizad el código del ejemplo encargado de la generación del gráfico SVG.

- Añadir dicha funcionalidad al ejemplo modificado de tipo *Bubble Chart* que se realizó en el tercer paso del boletín.

- Guardar el fichero en formato SVG y abrirlo mediante una programa de edición de gráficos vectoriales.