
IIC2026

Visualización de Información

— Hernán F. Valdivieso López —
(2022 - 2 / Clase 24)

Antes de empezar... Revisión de contenidos (RC)

No se olviden del control sobre redes y árboles.

- Tienen hasta el sub-siguiente martes para responderlos.

El **otro jueves** subiré la actividad de inventar pregunta de verdadero y falso II

Hoy no tendremos actividad RC.

Temas de la clase - Grafos y árboles en D3

1. Simulación de Fuerza
2. Árboles
3. PynetworkD3

Modalidad de esta clase

Híbrido entre teoría y programación.

1. Para cada tema se explicará lo que deseamos hacer junto con algunos de los comandos más importantes.
2. Luego se pasará a la etapa de programación donde haremos uso de dichos comandos.
3. Inspirándose en [Art Attack](#), se subió el código resultante al que “idealmente” llegaremos al final de la clase. A diferencia del código que haremos en clase, estos archivos además incluyen comentarios de casi cada línea del código.

Simulación de fuerza

Simulación de fuerza

d3.forceSimulation

- Objeto de D3 encargado de gestionar una simulación de fuerza entre una lista de nodos con el fin de determinar sus posiciones.
- Funciona a base de "ticks". En cada tick actualiza la posición de los nodos en función de las fuerzas definidas.
- Existen múltiples fuerzas a incluir en la visualización. En esta ocasión usaremos 4:
 - Fuerza producida por los enlaces
 - Fuerza de atracción o repulsión de nodos
 - Fuerza de colisión entre nodos
 - Fuerza de centro de gravedad.

Simulación de fuerza

d3.forceSimulation

- Existen múltiples fuerzas a incluir en la visualización. En esta ocasión usaremos 4:

```
const simulacion = d3.forceSimulation(nodos)

  .force("enlaces", d3.forceLink(enlaces).id((d) => d.nombre))

  .force("carga", d3.forceManyBody())

  .force("colision", d3.forceCollide(10))

  .force("centro", d3.forceCenter(width / 2, height / 2));
```

Simulación de fuerza

d3.forceSimulation

- La simulación funciona de forma interna con 2 valores:
 - **alpha**: valor para determinar qué tanto cambian las posiciones en el tiempo. A mayor valor, el cambio de posición entre un tick y otro es más grande. A medida que pasa la simulación, este número disminuye.
 - **alphaMin**: valor mínimo que debe tener **alpha** para realizar la simulación. Una vez que **alpha** es menor a **alphaMin**, la simulación se detiene.
- Mediante código podemos cambiar el valor de **alpha** y reiniciar la simulación.
`simulacion.alpha(0.8).restart()`

Simulación de fuerza

Vamos al código  

Simulación de fuerza

Links de interés

- [d3-force / D3 / Observable](#)
- [GitHub - d3/d3-force: Force-directed graph layout using velocity Verlet integration.](#)
- [D3 Force layout | D3 in Depth](#)

Árboles en D3

Árboles en D3

Datasets

Formato Jerárquico

- Cada nodo contiene, de forma anidada/recursiva, a sus hijos.
- Generalmente son guardados en formato JSON.
- Si un nodo no tiene hijos, su lista está vacía.
- Cada nodo puede contener más información sobre él.

```
{
  "nombre": "A",
  "hijos": [
    {
      "nombre": "B",
      "hijos": [
        {
          "nombre": "D",
          "valor": 30,
          "hijos": []
        },
        {
          "nombre": "E",
          "hijos": []
        }
      ]
    },
    {
      "nombre": "C",
      "hijos": []
    }
  ]
}
```

Árboles en D3

Datasets

Formato Tabular

- Cada fila representa un nodo.
- Generalmente son guardados en CSV.
- El nodo debe presentar un identificador único y un atributo que apunte al identificador del nodo padre.
- Cada nodo puede contener más información sobre él.

nombre, padre, valor

A, , 10

B, A, 23

C, A, 21

D, B, 54

E, B, 1

Árboles en D3

Procesar datos

d3.hierarchy

- Recibe un dataset en forma de diccionario.
- Se le debe indicar, mediante una función, cómo acceder a los hijos de cada nodo.

```
const raiz = d3.hierarchy(datos, (d) => d.hijos);
```

```
{
  "nombre": "A",
  "hijos": [
    {
      "nombre": "B",
      "hijos": [
        {
          "nombre": "D",
          "valor": 30,
          "hijos": []
        },
        {
          "nombre": "E",
          "hijos": []
        }
      ]
    },
    {
      "nombre": "C",
      "hijos": []
    }
  ]
}
```

Árboles en D3

Procesar datos

d3.stratify

- Recibe un dataset en forma de lista de nodos.
- Se le debe indicar, mediante atributos específicos y una función, cómo acceder al ID de cada nodo y al ID del padre.

```
const stratify = d3
  .stratify()
  .id((d) => d.nombre)
  .parentId((d) => d.padre);
```

```
const raiz = stratify(datos);
```

nombre, padre, valor

A, , 10

B, A, 23

C, A, 21

D, B, 54

E, B, 1

Árboles en D3

Procesar datos

d3.hierarchy y d3.stratify

- Ambos retornan el *dataset* procesado con estructura anidada.
- Tiene 3 métodos de utilidad:
 - `.descendants()` para obtener una lista de todos los nodos.
 - `.links()` para obtener una lista de los enlaces.
 - `.leaves()` para obtener una lista de todos los nodos hojas (aquellos sin hijos).

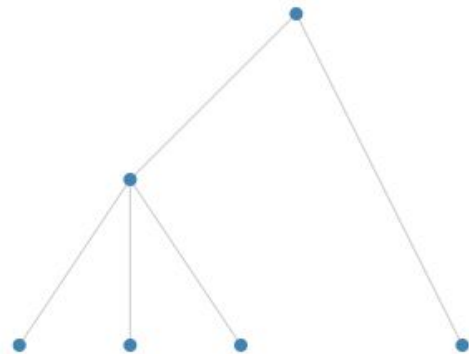
```
▼ Ed ⓘ 1_arbol_procesamiento.js:28  
  ▶ children: (2) [Ed, Ed]  
  ▶ data: {nombre: 'A', padre: null, valor: 10}  
    depth: 0  
    height: 4  
    id: "A"  
    parent: null  
  ▶ [[Prototype]]: Object
```


Árboles en D3

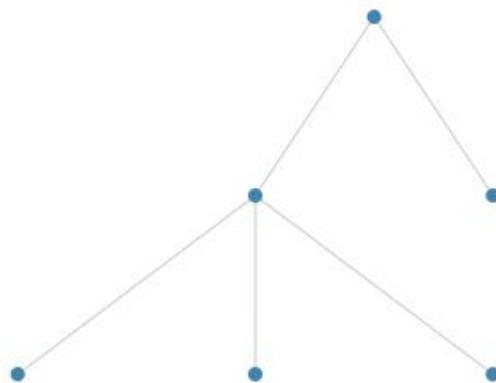
Métodos para generar las visualizaciones

`d3.cluster` y `d3.tree`

- Posiciona los elementos considerando una visualización nodo-enlace.
- Calculan las posiciones x,y de cada nodo y enlace para reflejar visualmente la jerarquía.
- La principal diferencia es que `d3.cluster` posiciona todos los nodos hoja a la misma profundidad.



`d3.cluster`



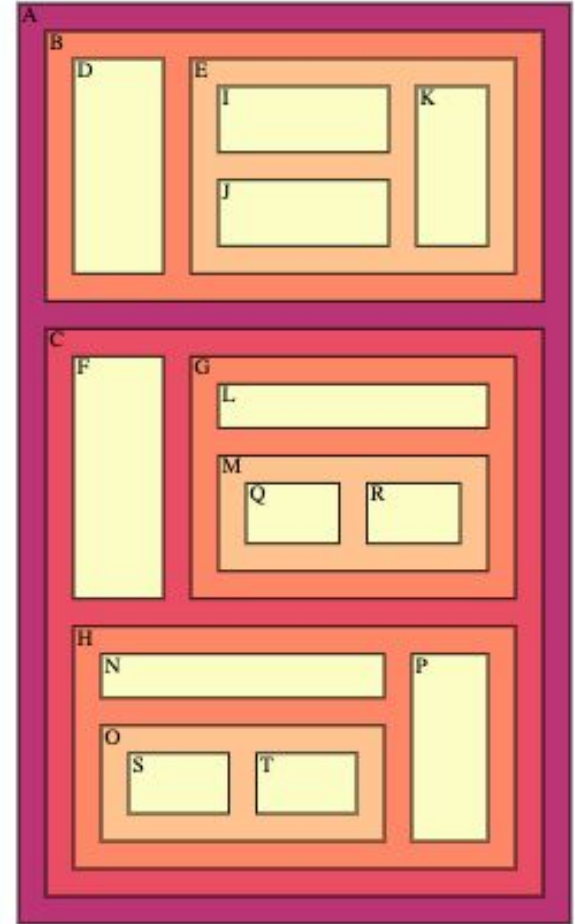
`d3.tree`

Árboles en D3

Métodos para generar las visualizaciones

d3.treemap

- Posiciona los elementos considerando una visualización de treemap.
- Calculan las posiciones x,y de cada nodo y el tamaño de cada rect.
- Cada nodo debe tener un atributo **value** para determinar el tamaño a ocupar.

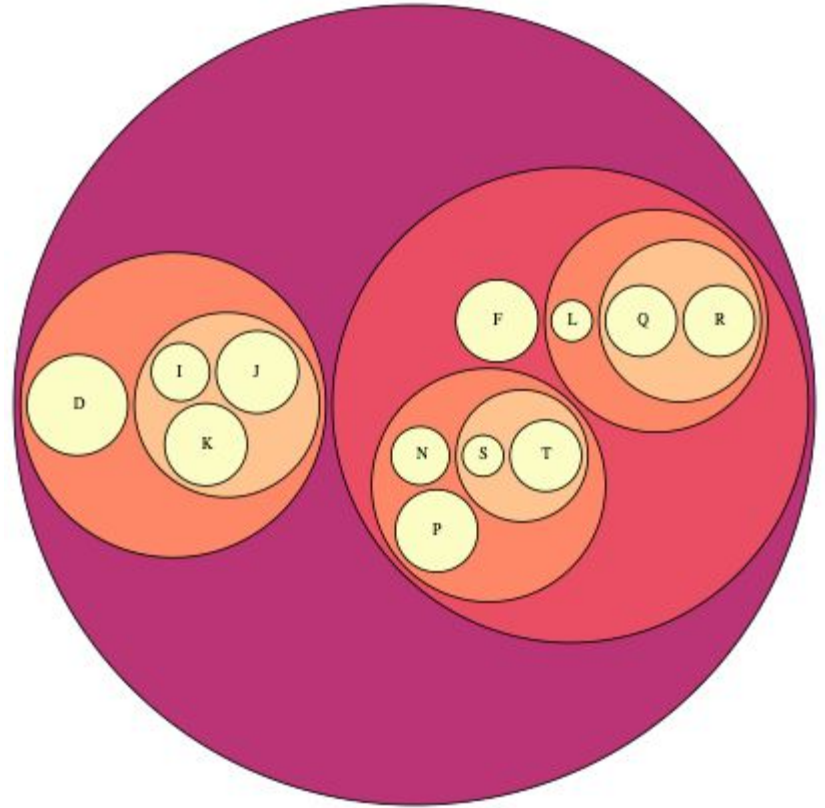


Árboles en D3

Métodos para generar las visualizaciones

d3.pack

- Posiciona los elementos considerando una visualización de *circle packing*.
- Calculan las posiciones x,y de cada nodo y el tamaño de cada circle.
- Cada nodo debe tener un atributo **value** para determinar el tamaño a ocupar.



Árboles en D3

Vamos al código  

Árboles en D3

Links de interés

- [d3-hierarchy / D3 / Observable](#)
- [GitHub - d3/d3-hierarchy: 2D layout algorithms for visualizing hierarchical data.](#)
- [D3 Hierarchies | D3 in Depth](#)

PyNetworkD3

<https://pypi.org/project/pynetworkd3/>

PyNetworkD3

- Problema
 - Actualmente hay pocas librerías que permiten visualizar redes en Python.
 - La principalmente utilizada hace visualizaciones estáticas y muy toscas.
 - Modificar visualizaciones existentes requiere muchas veces modificar código base.
- Solución
 - <https://github.com/Hernan4444/PyNetworkD3>
 - Librería para confeccionar visualizaciones de redes en Python de forma más simple.
 - Dado que construye código en D3, permite modificarlas sin necesidad de acceder al código base de la librería.

Próximos eventos

Martes

- Feriado, no hay clases 🎉🎊

Miércoles

- Se entrega la T03 y no hay ayudantía planificada. Se usará para responder dudas de último momento sobre la tarea.
- Se está haciendo el máximo esfuerzo para tener el enunciado del examen y así ya sepan (1 semana antes) qué se viene.

Jueves

- Veremos privacidad de datos y hablaremos del examen.

IIC2026

Visualización de Información

— Hernán F. Valdivieso López —
(2022 - 2 / Clase 24)
