Aula 5- Layouts

Resumo

- Force layout
- Trees
- Tree Map

Nota:
Este tutorial foi realizado usano a versão 4 do d3.

O d3 permite realizar de uma forma simples gráficos baseados nos *layouts* seguintes: *Networks, Chord, Cluster, Force, Hierarchy, Histogram, Pack, Partition, Pie, Stack, Tree, Treemap*. Nesta aula vamos explorar alguns destes *layouts*. Pode explorar alguns dos layouts do d3 em http://d3indepth.com/layouts/ .

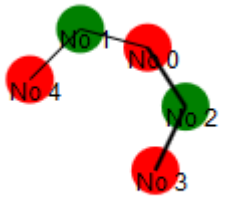
5.1 Force layout

O *Force Layout* controla um algoritmo de física que define a posição dos nós e utiliza um temporizador (evento de *tick*) permitindo variar as posições dos nós e outros parâmetros do gráfico ao longo do tempo. No link http://d3indepth.com/force-layout/ é possível encontrar alguma informação adicional sobre o tipo de forças possíveis de usar.
Teste o exemplo fornecido (D3_5_1.htm). Note que este lê os dados do ficheiro nos.json. Modifique o ficheiro par acrescentar um nó na visualização.

Modifique o exemplo fornecido para apresentar o nome do nó em cima do mesmo (pode aproveitar o código para colocação de texto do exemplo 1.7)
Modifique ainda o ficheiro de dados associando um grupo a cada nó e um peso a cada ligação (escolhe valores entre 1 e 2):

```
{ "name": "No 0", "group":1 },
{ "source": 0, "target": 1, "value":1 },
```

Utilize essa informação para mapear grupos diferentes com cores diferentes e alterar a largura das ligações em função do seu peso obtendo algum parecido com a figura seguinte:



Para tornar o gráfico mais interativo, pode utilizar as opções de drag:
Adicione o código seguinte no "node":

```
            .call(d3.drag()
              .on("start", dragstarted)
              .on("drag", dragged)
              .on("end", dragended));
```

E as funções de "drag":

```
            function dragstarted(d) {
        if (!d3.event.active) simulation.alphaTarget(0.3).restart();
        d.fx = d.x;
        d.fy = d.y;
    }

    function dragged(d) {
        d.fx = d3.event.x;
        d.fy = d3.event.y;
    }

    function dragended(d) {
        if (!d3.event.active) simulation.alphaTarget(0);
        d.fx = null;
        d.fy = null;
    }
```

Veja o que ocorre. Pode também testar o exemplo com dados mais extenso usando o ficheiro miserables.json. Altere a visualização assim obtida a seu gosto, modificando por exemplo os valores de inicialização do layout. Pode ainda ver a influência dos vários parâmetros associados a força (http://d3indepth.com/force-layout/) .

5.2 Tree

Crie agora um *tree layout* a para visualizar a hierarquia do DETI (deti.json). Comece por criar a hierarquia usando o código seguinte.

```
var root = d3.hierarchy(data);
```

Pode agora definir uma árvoreque utiliza a hierarquia calculda com o código seguinte:

```
var treeLayout = d3.tree()
.size([400, 180]);

treeLayout(root);
```

Por fim visualize os nós e as ligações das árvores usando o código seguinte para aceder ao nós e ligações da hierarquia. Para visualizar os nós na posição correta utilize os valores d.x e d.y dos "descendants" que retornam as posições dos nós.

Relativamente as linhas, deve utilizar para os attributos (x1,y1) e (x2,y2) das linhas as coordenadas dos nós fontes (source.x e source.y) e destino (target.x e target.y). Altere ainda a cor e tamanho das linhas e círculos ao seu gosto.

```
// Nodes
var nodes = svg.selectAll('circle')
  .data(root.descendants())
  .enter()
  .append('circle');

// Links
var links = svg.selectAll('line')
  .data(root.links())
  .enter()
  .append('line');
```

Pode opcionalmente adicionar texto junto aos nós acedendo ao atributo d.data.name. Note que a hierarquia retorna coordenadas com o valor 0, se quiser movimentar o gráfico para outra posição deve acrescentar transformações. Pode observar como isso é feito através da criação de grupos no SVG no exemplo em: http://d3indepth.com/layouts/.

5.3 Tree map

Crie agora um *treemap layout* a partir do ficheiro DETI.json com a estrutura de ensino do departamento. Utilize o código seguinte para criar uma estrutura que permita represnetar uma hierarquia.

```
var root = d3.hierarchy(data)
```

Comece por criar o layout dentro da função de desenho chamada pela função `d3.json`:

```
var treemapLayout = d3. treemap()
       .size([width, height])
       .paddingOuter(20);
```

Associe agora a hiearaquia ao treemap:

```
treemapLayout(root);
```

Pode agora criar os nós da árvore usando o código:

```
var cells = svg.selectAll('rect')
  .data(root.descendants())
  .enter()
  .append("rect")
```

Relativamente aos retângulos, deve utilizar para os attributos (d.x0,d.y0) e (d.x1,d.y1) das coordenadas superiores esquerdas e inferiores direitas para desenhar os mesmos.

Para visualizar o treemap, deve representar os retângulos com alguma opacidade. Pode utilizar o estilo seguinte por exemplo:

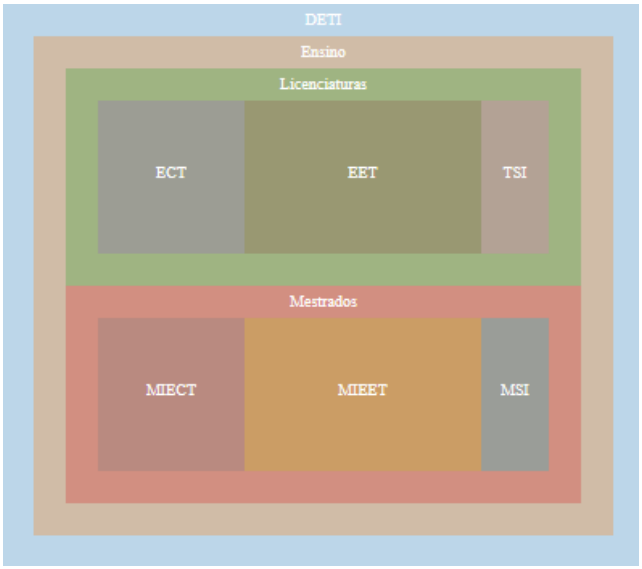
```
<style>
rect {
	fill: cadetblue;
	opacity: 0.3;
	stroke: white;
}
</style>
```

Para especificar uma cor diferente para os nós, pode usar o código seguinte onde color é uma escala ordinal que mapeia 10 valores ordinais para cores.

```
var color = d3.scaleOrdinal(d3.schemeCategory10);

.attr("fill", function (d) {  return color(d.data.name); })
```

Acrescente ainda no gráfico um texto centrado nos nós folhas (sem filhos - (d.children)) e na parte superiori dos outros nós.com o nome dos para obter algo semelhante a figura seguinte.



Modifique o gráfico ao seu gosto e teste diferentes opções para calcular as posições dos nós (ver em http://d3indepth.com/layouts as opções possíveis para o mode). Qual a opção usada por omissão?