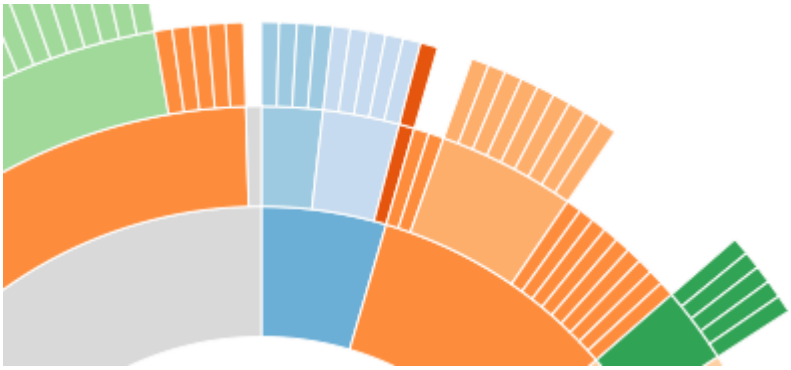
- 如发现翻译不当或有其他问题可以通过以下方式联系译者:
- 邮箱：zhang_tianxu@sina.com
- QQ群：D3数据可视化205076374，大数据可视化436442115

分区布局将会产生邻接的图形：一个节点链的树图的空间填充转化体。节点将被绘制为实心区域图（无论是弧还是矩形），而不是在层次结构中绘制父子间链接，节点将被绘制成固定区域（弧度或者方形），并且相对于其它节点的位置显示它们的层次结构中的位置。节点的尺寸将编码为一个量化的维度，这将难以在一个节点链图中展示。



就像D3的其他类，布局将遵循方法链模式，setter方法返回布局本身，并通过一个简洁的语句来调用多个setter方法。

d3.layout.**partition**()

创建一个新的带有默认设置的分区布局：默认为值降序的排序顺序；默认值的访问器假定为每一个输入值是一个带有数值属性的对象；默认子值的访问器假定为每一个输入值是一个带有子值数组的对象，大小为1为1×1。

partition(*root*)
partition.**nodes**(*root*)

运行分区布局，将返回指定根节点 *root* 的相关联的节点阵列数组。分区布局是D3家族(hierarchical layouts)中的一部分。这些布局将遵循相同的基本结构：传递给布局的输入参数值是层次结构的根节点，并且输出返回值将是一个代表所有节点经过计算的位置的数组。每个节点上将拥有以下几个属性：

- `parent` - 父节点，或空的根节点。
- `children` - 子节点的阵列数组，或者为空的叶节点。
- `value` - 该节点的值，值访问器所返回的值。
- `depth` - 节点的深度（即节点的层级数），根节点为0。
- `x` - 节点位置的最小*x*坐标。
- `y` - 节点位置的最小*y*坐标。
- `dx` - 节点位置的*x*范围。
- `dy` - 节点位置的*y*范围。

虽然布局只有一个*x*和*y*尺寸，但它却可以表示一个任意的坐标系；例如，你可以把*X*作为半径、*y*作为角度用以产生径向而非笛卡尔布局。在笛卡尔取向上，*x*，*y*，*dx*和*dy*分别相当于SVG矩形元素(rect)的“x”，“y”，“width”和“height”属性。在径向取向上，它们可以被用于计算弧(arc)生成器的innerRadius，startAngle，outerRadius和endAngle。笛卡尔取向可被称为**冰柱树(icicle tree)，**而径向取向被称为**旭日图(sunburst)**。

partition.**links**(*nodes*)

给定指定的节点数组 *nodes*，比如由nodes函数返回的节点，将返回一个对象数组，该数组表示从父到子的每个节点的链接。叶节点将不会有任何的链接。每个链接是一个对象，每个对象具有以下两个属性：

- source - 父节点（如上所述）。
- target - 子节点。

此方法适用于检索一组适于显示的链接描述，通常与对角线(diagonal)形状生成器结合使用。例如：

```
svg.selectAll("path")
    .data(partition.links(nodes))
  .enter().append("path")
    .attr("d", d3.svg.diagonal());
```

partition.**children**([*children*])

如果指定了*children*，将设置指定的children访问器方法。如果没有指定*children*，则返回当前的children访问器方法，默认输入数据是一个带有一个children数组的对象：

```
function children(d) {
  return d.children;
}
```

通常情况下，使用d3.json可以很方便地装入节点层次结构，并且将输入层次结构作为嵌套JSON对象。例如：

```
{
 "name": "flare",
 "children": [
  {
   "name": "analytics",
   "children": [
    {
     "name": "cluster",
     "children": [
      {"name": "AgglomerativeCluster", "size": 3938},
      {"name": "CommunityStructure", "size": 3812},
      {"name": "MergeEdge", "size": 743}
     ]
    },
    {
     "name": "graph",
     "children": [
      {"name": "BetweennessCentrality", "size": 3534},
      {"name": "LinkDistance", "size": 5731}
     ]
    }
   ]
  }
 ]
}
```

children访问器将在层次结构根节点中首先被调用。如果访问器返回null，则该节点被认为为叶节点，直到布局遍历终止。否则，访问器就会返回一个表示子节点的数据元素的数组。

partition.**sort**([*comparator*])

如果指定的*comparator*，将使用指定的比较方法来设置为布局的同级节点的排序顺序。如果没有指定*comparator*，返回当前组的排序顺序，其默认为相关的输入数据的数值属性降序排列：

```
function comparator(a, b) {
  return b.value - a.value;
}
```

比较器方法用来被节点对调用，此节点对传递给每个节点的输入数据。若比较器方法为null，则排序将被禁用，之后将使用树的遍历顺序。比较器方法也可以使用d3.ascending 或 d3.descending实现。

partition.**value**([*value*])

如果指定*value*，便将其设置为指定的访问器方法。否则返回当前值访问器，即输入数据是一个带有数值属性的对象：

```
function value(d) {
  return d.value;
}
```

值访问器将被每个输入的数据元素调用，并且一定会返回一个数字，该数字表示该节点的数值属性值。此值将被成比例的设置每个节点的面积值。

partition.**size**([*size*])

如果指定的size，将通过指定的二元素数组[x,y]设置为有效布局的大小:

```
var width = 1800, height = 2000;
var partition = d3.layout.partition()
    .size([width, height])
    .value(function(d) { return d.size; });
```

如果没有指定size，返回当前布局大小，默认为1×1。

以下为新增内容

```
Notice, that if the size is not explicitly specified, i.e. defaults to 1×1, the
calculated coordinates for each element will be normalized, i.e. each coordinate will
be < 1, whereas the sum of all the coordinates giving 1. Consequently, you will not
see anything on UI ( since 1x1 is too small to see anything). A good way to use such a
situation is to apply linear range scaling. The rendering result will be the same as
in case of specifying explicitly the size of the diagram, however you will get a more
flexible solution for further implementations:
```

```
// first, declare scaling for X and Y axis
 var width = 1800, height = 2000,
 x = d3.scale.linear().range([0, width]),
 y = d3.scale.linear().range([0, height]);

// then apply the scaling for each calculated coordinate
svg.selectAll("rect").data(nodes)
            .enter().append("svg:rect")
            .attr("x", function(d) { return x(d.x);})
```

```
            .attr("y", function(d) { return y(d.y);})
            .attr("width", function(d) {  return  x(d.dx);})
            .attr("height", function(d) { return y(d.dy); });
```

以上为新增内容

- Harry 译 2014-11-29
- 咕噜校对 2014-11-30 21:02:50