- 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

矩形树图最先由Ben Shneiderman在1991年提出; 矩形树图会递归的对一块矩形区域进行切分, 以达到层级展示的效果. 正如分区布局中, 每个节点的大小都是显而易见的. 正方化的矩形树图使用近正方的矩形, 因此, 相比于传统的切块或切片图, 具有更好的可读性和节点大小易读性. 还有其他一些关于矩形树图的算法, 比如: Voronoi 和 jigsaw, 但是并不常用.

和其他 **D3** 类一样, 布局也支持链式调用.

链式调用, 即: 一个setter方法会返回当前实例, 意味着意味着一个setter方法调用后可以紧接着另一个setter方法的调用, 如: selection.attr('x', 1).style('color', '#999')

d3.layout.**treemap**()

使用默认的配置实例化一个新的矩形树布局: 默认的递减排序, 默认的值存取器会假定输入的数据是一个有value属性的对象, 默认的子节点存取器会假定输入的数据是一个以对象为元素的数组, 默认的绘图区域大小是[1, 1].

treemap(root)

treemap.nodes(root)

运行树形图布局，返回与指定根节点相关的节点数组。树状图布局是D3族分层布局的一部分。这些布局遵循相同的基本结构：布局的输入参数是层次结构的根节点，输出的返回值是代表所有节点计算的位置的数组。每个节点还有一些属性：

• parent -父节点，或根是null。 • children -子节点的数组，或叶节点是null。 • value -该节点的值，通过值访问器返回。 • depth -节点的深度，根开始为0。 • x -节点位置的最小横坐标。 • y -节点位置的最小纵坐标。 • dx -节点位置的x轴宽。 • dy -节点位置的y轴宽。

请注意，这会改变传递的节点！ 虽然布局有x 和 y尺寸，这表示一个任意的坐标系统；例如，你可以以x为半径y为角度产生辐射状而不是直角的布局。在直角方向，x，y，dx和dy对应于“x”，“y”，“宽度”和“高度”的SVG矩形元素的属性。

treemap.links(nodes)

给定指定的节点数组，如返回的节点，返回一个对象数组代表从父到子节点的链接。叶节点将不会有任何链接。每一个关系都有两个属性的对象： • source -父节点（如上所述）。 • target -子节点。

这种方法是有用的，用于检索一组适合于显示的链接描述，经常与对角线形状发生器一起使用。例如：
svg.selectAll("path") .data(partition.links(nodes)) .enter().append("path") .attr("d", d3.svg.diagonal());

treemap.children([children])

如果指定了子节点，设置指定子节点访问器函数。如果未指定子节点，返回当前子节点访问器函数，默认情况下假定输入数据是一个对象，一个子节点数组： function children(d) { return d.children; }

通常，使用d3.json很方便加载节点的层次结构，并用嵌套JSON对象代表输入层次结构。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}] }] }] }

子节点的访问首先在层次结构的根节点调用。如果访问器返回null，则该节点在布局遍历终止被认为是叶节点。否则，访问应返回表示子节点的数据元素的数组。

treemap.sort([comparator])

如果比较器是指定的，使用指定比较函数设置布局的兄弟节点的排序顺序。如果没有指定比较器，返回当前组的排序顺序，默认为通过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; }

比较器函数节点对被调用，传递每个节点的输入数据。空比较器禁用排序并使用树的遍历顺序。比较器的功能也可以用d3.ascending或d3.descending实施。

treemap.value([value])

如果指定了值value ，将值访问器为指定的函数。如果值是未指定的，则返回当前值的访问器，假定输入数据是一个带有数值属性的对象： function value(d) { return d.value; } 值访问器调用被每个输入数据元素，并且必须返回一个表示该节点的数值。这个值用于按比例设定每个节点面积的值。

treemap.size([size])

如果尺寸size 是指定的，设置现有布局尺寸，指定代表x和y的数值的两元素数组。如果尺寸不是指定的，则返回当前大小，默认为1×1。

treemap.padding([padding])

为每个树形图单元获取或设置填充，以像素为单位。填充确定保留父节点和子节点之间的额外空间量；这个空间可以用于通过外边缘表示层次结构，或保留父标签的空间。如果不使用填充，那么树中叶子的内容会完全填满布局的大小。 如果指定了填充，则设置新的填充并返回树形布局；如果没有指定填充，则返回当前的填充。填充可以指定几种方法：

• 空值禁用填充；空等同于零。 • 一个数字表示均匀填充，以像素为单位，在所有四面。 • 数组的数值表示上，右，下和左填充值。 填充也可以被指定为一个函数，它返回上面的三个值中的一个。这个函数为每个内部（非叶）节点求值，并且可以被用来动态地计算填充。

treemap.round([round])

如果四舍五入round 是指定的，设置树状图布局是否将全面取整到像素边界。这可以很好地避免SVG边缘有锯齿。如果四舍五入不是指定的，返回是否将树状图四舍五入。

treemap.sticky([sticky])

如果粘滞sticky 是指定的，设置树形布局是否是“粘滞”： 粘滞树形布局将保留整个过渡中节点的相对排列。节点分成正方化水平和垂直行，通过每行中最后一个元素存储一个z属性持续整个更新；这允许节点平滑地调整大小，没有将阻

碍感知变化值的洗牌或闭塞。但请注意，这会为两种状态之一生成一个次优的布局。如果没有指定粘滞，则返回的树形布局是否已被指定粘滞。

实现注意事项：粘滞树形图的内部缓存节点数组；因此，相同的布局实例不可能再使用于多个数据集。当切换带有粘滞布局的数据集时重置缓存状态，再次调用sticky(true)。自从版本1.25.0，层次布局不再在每次调用中默认复制输入数据，因此它可能是能够消除高速缓存和使布局完全无状态。

treemap.mode([mode])

如果模式mode 是指定的，设置布局的算法。如果模式没有指定，返回当前的布局算法，默认为“squarify”。支持以下模式： • squarify - rectangular subdivision; squareness controlled via the target ratio. • slice - horizontal subdivision. • dice - vertical subdivision. • slice-dice - alternating between horizontal and vertical subdivision. • squarify -矩形细分；矩形通过目标比率控制。 • slice –水平细分。 • dice -垂直细分。 • slice-dice -水平和垂直细分之间的交替。

- ?译gulu
- 校对2014-12-7 08:43:59