- 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

堆叠布局需要一个二维的数据数组，并计算基准线；这个基准线会被传到上层，以便生成一个堆叠图。支持多个基线算法，以及启发式的排序算来可以提高感知灵敏度，就像拜伦(Byron)和瓦腾伯格(Wattenberg)在"Stacked Graphs—Geometry & Aesthetics"（http://www.leebyron.com/else/streamgraph/download.php?file=stackedgraphs_byron_wattenberg.pdf）中所说的那样。

堆叠布局可以工作在任意的二维x，y坐标系空间，就像是D3的其他布局一样，包括树布局(tree)。因此，图层可以被垂直、水平叠放，或者是其他径向的叠放(radially)。尽管图表的默认偏移是零，但是依然可以使用扭动或摆动(wiggle)的偏移量来绘制流图，它会尽量的减少在偏移时所产生的锯齿边界。

d3.layout.stack()

构造一个新的堆叠布局，使用默认的偏移（零）和排序（空null）；返回的布局对象是一个对象也是一个函数；也就是说：你可以向调用函数一样调用布局，布局是具备改变其行为的方法的；和D3中其他类相似，布局遵循方法链模式，其中setter方法返回布局本身，并允许在一个简单语句中调用多个setter方法。

stack(layers[, index])

为各层计算y坐标的基线，并传到相应的层中。最简单的情况，层是一个二维的值数组；所有的第二维的数组必须是相同的长度。y和x访问器被用来分别定义每层的x坐标位置的y方向厚度；因此下面这些值都是必须的： • x - x位置处所对应的值，即x坐标； • y - y处厚度所对应的值； • y0 - y方向最小的y值，即基线； 以上这些属性可以通过重写访问器或out函数进行自定义。

stack.values([accessor])

指定在每一层中如何从相关联的元素中提取值，访问器是一个函数，并被传递给层以被调用在每一个输入层上，相当于是在计算堆叠布局前调用了layers.map(accessor)。默认的值函数是内置对象，类似于标识函数。如果未指定访问器accessor，则返回当前的值访问器。 值访问器可以被用于关联每层额外的数据，而不仅仅是每一个点；例如，假设你的数据结构如下： var layers = [{ "name": "apples", "values": [{ "x": 0, "y": 91}, { "x": 1, "y": 290}] }, { "name": "oranges", "values": [{ "x": 0, "y": 9}, { "x": 1, "y": 49}] }]; 指定一个值访问器来检索每层的点： var stack = d3.layout.stack() .offset("wiggle") .values(function(d) { return d.values; }); 然后，如果你想给每层添加一个tooltip，你可以这样写： svg.selectAll("path") .data(stack(layers)) .enter().append("path") .attr("d", function(d) { return area(d.values); }) .append("title") .text(function(d) { return d.name; });

stack.offset([offset])

如果指定offset，则设置堆叠的偏移算法为指定的offset ； 如果未指定offset ，则返回当前的offset；以下的字符串值可以被使用： • silhouette - 居中流，类似于 ThemeRiver； • wiggle - 尽量减少斜率变化比例； • expand -标准化层以填补在范围[0,1]之间； • zero - 使用零基线，即y轴.； 另外offset 可以是一个函数；输入到该函数的是层数据；并已被标准化显示：一个二维的值数组，每个元素被表示为一个二元素的数组[x,y]；函数的返回值必须是一个表示基线y坐标的数组。例如，默认的零偏移实现如： function offset(data) { var j = -1, m = data[0].length, y0 = []; while (++j < m) y0[j] = 0; return y0; }

stack.order([order])

如果指定order，设置堆叠的排序为指定的order；如果未指定，则返回当前的order。以下字符串类型的值可被使用：• inside-out - 通过最大值的索引进行排序，然后使用平衡加权； • reverse - 反转输入层的次序； • default - 使用输入层顺序； 另外order也可以是函数；输入到该函数的是层数据，并已被标准化显示：一个二维的值数组，每个元素被表示为一个二元素的数组[x, y]；该函数的返回值必须是一个表示层顺序的索引数组。例如，默认的顺序实现如：function order(data) { return d3.range(data.length); } 参见 d3.range.

stack.x([accessor])

指定如何访问每个值位置中的x坐标。如果指定accessor，则设置访问器为指定的函数；如果未指定accessor，则返回当前绑定的访问器函数；在默认情况下假定每个输入值都有x属性： function x(d) { return d.x; } X访问器会被每一个输入值调用，并且每一层，被传递的参数有当前数据(d)和数据元素索引(i)；返回值必须是一个数字；虽然x访问器会被每层调用，堆叠布局是假定所有层的x坐标是一致的；也就是说，堆叠布局目前需要每层都是均匀的，在相同的x坐标下，都必须包含相同个数的值；如果你的数据不符合这种规则，在你进行堆叠布局前需要整理好数据符合这种规则。

stack.y([accessor])

指定如何访问每个值位置中的y坐标。如果指定accessor，则设置访问器为指定的函数；如果未指定accessor，则返回当前绑定的访问器函数；在默认情况下假定每个输入值都有y属性： function y(d) { return d.y; } X访问器会被每一个输入值调用，并且每一层，被传递的参数有当前数据(d)和数据元素索引(i)；返回值必须是一个数字；随着异常的扩大偏移，堆叠布局不会执行任何的数据自动缩放；为了简化缩放，可以使用关联的线性缩放linear scale或其他类似的。

stack.out([setter])

指定如何传递计算的基线给上层；如果指定setter，它将被用作该输出/传递功能上；如果未指定setter，则返回当前的setter；默认情况下假定每个输入值都有y和y0属性： function out(d, y0, y) { d.y0 = y0; d.y = y; } setter会被每一个输入值调用，并且每一层，被传递的参数有当前数据(d)，已计算的y0值和已计算的y厚度；在除了expand偏移的所有情况下，y厚度是相同于y的返回值，因此可被忽略。

- 魏飞译 2014.07.08.10.04
- gulu 校对 2014-12-6 23:08:08