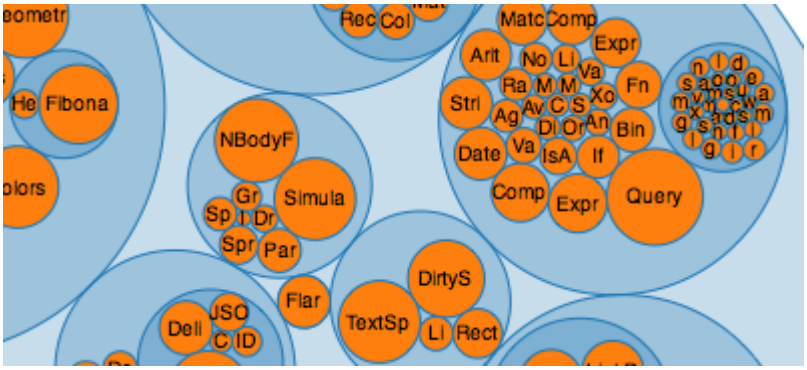
> *Wiki* ‣ *[[API--中文手册]]* ‣ *[[布局]]* ‣ ***包布局***

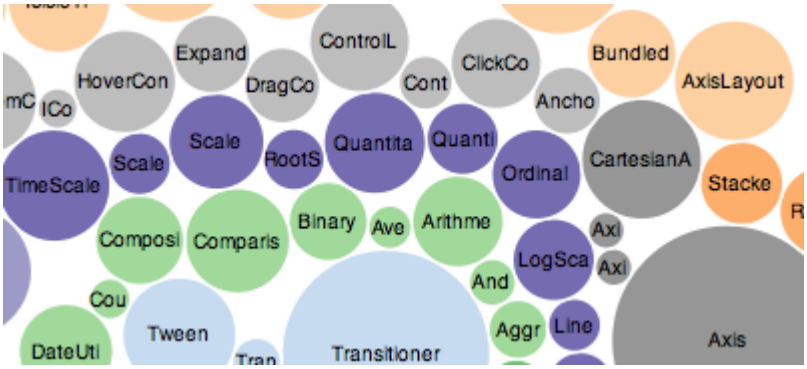
- 如发现翻译不当或有其他问题可以通过以下方式联系译者:
- 邮箱：zhang_tianxu@sina.com
- QQ群：D3数据可视化205076374，大数据可视化436442115

下图采用包含（嵌套）来展现层级结构。每一个叶子节点的大小都显示了每个数据点数量的大小。如图所示，通过小圈的大小累积逐渐接近于一个大圈，但要注意的是，由于空间上的浪费，在层级之间也会存在一些失真，但只要叶子节点可以精确的比较就行。尽管圆形填充(circle packing)没有像treemap那样，高效率的使用了空间，空间上的浪费反而清晰地展现了分层效果。

此布局基于 Wang 等人的 Visualization of large hierarchical data by circle packing.



通过展开层级，**pack layout** 也可以被用来创建bubble charts（气泡图表）。



和D3中其他类相似，layout中的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。

d3.layout.**pack**()

使用默认设置创建一个新的包布局，默认的排序顺序是按值升序排序；默认的子访问器，假设每一个输入数据都是一个带有子数组的对象，默认的大小是1×1。

pack(*root*)
pack.**nodes**(*root*)

运行包布局，返回与指定根节点root相关联的节点的数组。簇布局是D3家族层次布局(hierarchical layouts)中的一部分。这些布局遵循相同的基本结构：将输入参数传递给布局是层次的根节点root ，输出的返回值是一个代表所有节点计算出的位置的数组。每个节点上填充以下几个属性：

- `parent` - 父节点，或根节点为null。
- `children` - 子节点数组，或叶子节点为null。
- `value` - 节点的值，作为访问器返回的值。
- `depth` - 节点的深度，根节点从0开始。
- `x` - 计算的节点位置的*x*坐标。
- `y` - 计算的节点位置的*y*坐标。
- `r` - 计算的节点半径。

pack.**links**(*nodes*)

给定一个特定节点的数组*nodes*，例如由nodes返回的节点，返回表示每个节点的从父节点到子节点链接的对象数组。而叶子节点将不会有任何的链接。每个链接都是一个对象，且具有两个属性：

- source - 父节点（正如上述描述的那样）。
- target - 子节点。

此方法检索一组适合展示的链接描述很有用，通常与对角线(diagonal)形状发生器一起用。例如：

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

pack.**children**([*children*])

如果指定 *children* 参数，则设置特定的子节点访问器函数。如果 *children* 没有指定，则返回当前子节点访问函数，默认假定输入的数据是一个带有子节点数组的对象：

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

通常情况下，可以使用d3.json很方便加载节点层次结构，代表输入层次结构的嵌套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}
     ]
    }
   ]
  }
 ]
}
```

子节点的访问器是第一次被层次中的根节点调用。如果存取器返回 `null`，则该节点被假定为叶节点并且布局遍历终止。否则，访问器应返回表示子节点的数据元素的数组。

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

如果 *comparator* 比较器指定了，则使用特定的比较器函数，设置同级节点布局的排序顺序。如果 *comparator* 没有指定，则返回当前组排序顺序，默认为升序排列，按通过相关的输入数据的数值属性 `value` 来排序：

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

这个比较器的函数被节点对调用，输入的数据传递给每个节点。空比较器禁止排序，使用树遍历的顺序。比较器函数也可以使用 d3.ascending 或 d3.descending 来实现排序。

pack.**value**([*value*])

如果 *value* 值指定了，则设置值访问器为指定的函数。如果 *value* 值未指定，则返回当前的值访问器，其中假定输入数据是一个带有一个数字值属性的对象：

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

对每一个输入的数据元素调用值访问器，并且必须返回一个数字，来表示该节点的数值。此值被用来按比例设置每个圆的面积为value。但是请注意，圆圈大小是严格地只在叶节点之间才存在可比性；内部的节点无法准确地比较，因为在打包子圆圈和他们父节点之间还留有空白。

pack.**size**([*size*])

如果指定了大小*size*，则设定可用的布局大小为指定的表示*x*和*y*的二元数组。如果*size* 大小没有指定，则返回当前大小，默认为1×1。

pack.**radius**([*radius*])

如果指定了半径*radius*，则设置半径的函数，用于计算每个节点的半径。如果半径*radius*为 `null`，因为默认情况下，半径自动从节点值确定，且调整为适合的布局大小。如果半径*radius*未指定，则返回当前的半径函数，默认为null。此半径*radius* 也可为均匀的圆大小指定为一个恒定数目。

pack.**padding**([*padding*])

如果指定*padding* ，则设置相邻圈之间的大概填充，以像素为单位。如果没有指定*padding* ，则返回当前的填充，默认为0。

- 何凯琳译 2014-12-01
- gulu校对2014-12-6 21:32:15