



下载APP



## 35| 设计（一）：如何让可视化设计更加清晰？

2020-09-21 月影

跟月影学可视化

[进入课程 >](#)



讲述：月影

时长 11:39 大小 10.68M



你好，我是月影。

在实际的可视化项目中，我们经常会遇到一种情况：用户期望所有的可视化图表都是简单明了的。实际上，这是不现实的。

因为我们拿到原始数据之后，第一步是分析数据，也就是从各种不同的角度尝试去观察数据，确定我们希望用户了解的信息。这些信息如果是简单清晰的，那么可视化结果就是简单直观的。如果用户想要了解的数据规律本身就很复杂，那么可视化图表所能做的事情也只能是尽可能清晰地展现用户关注的重要信息，屏蔽干扰信息，来降低用户理解数据的难度。



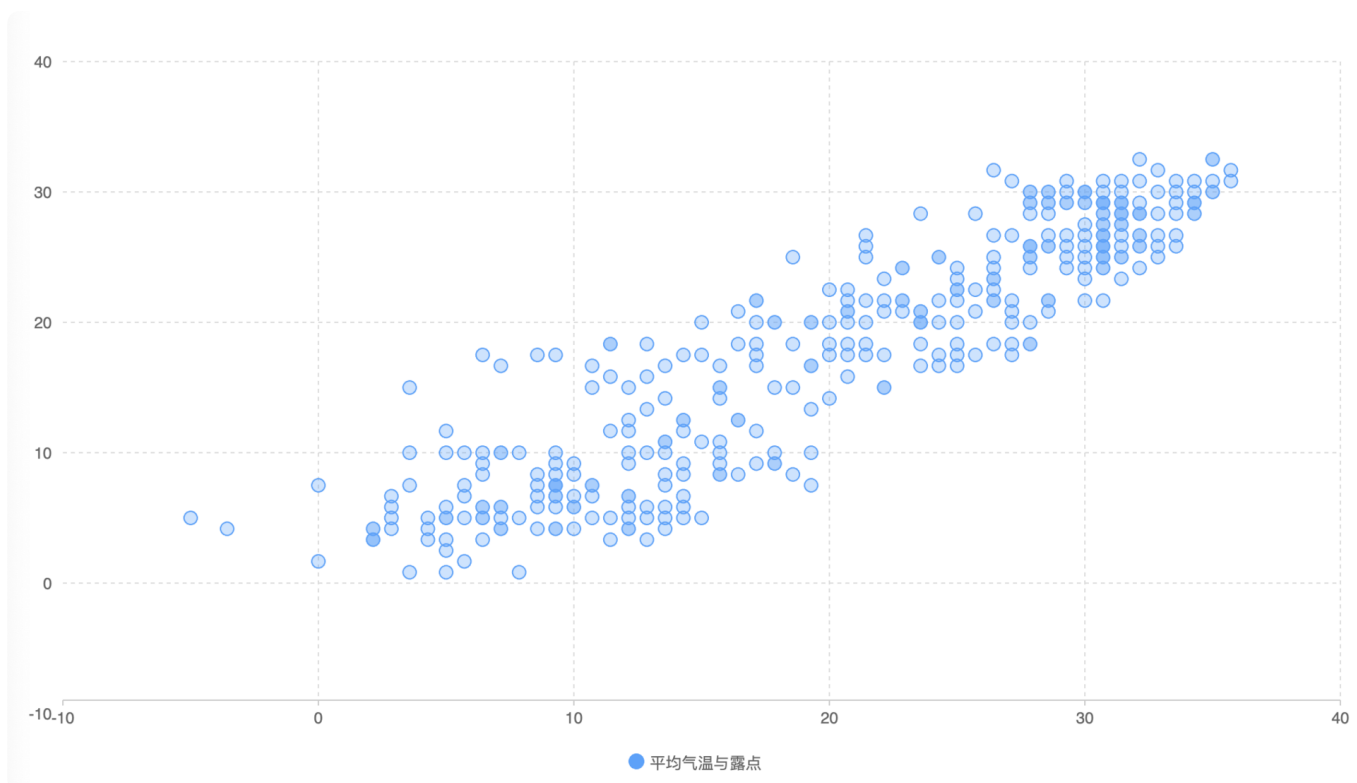
因此，我们要明白，在任何时候，制作可视化图表都是为了帮助人们理解抽象的数据，不管这些数据多复杂，都要尽可能让读者快速理解。如何才能做到这一点呢？简单来说，就是你要学会了解人们是怎样看数据的，然后将数据呈现得符合人们的思维和理解习惯。

接下来，我们就通过几个例子来学习一下，都有哪些方法可以轻松地把人们的注意力集中在数据信息上。

## 分清信息主次，建立视觉层次

我们可以先想这么一个问题：第一次看图表的时候，你都会注意哪些信息？如果是我的话，我总会试图在图表上找到什么有趣的东西。实际上，在看任何东西的时候，我们的眼睛总是倾向于识别那些引人注目的东西，比如，明亮的颜色，较大的物体，有特点的符号等等。因此，我们做可视化的时候，应当用醒目的颜色突出显示数据，把被淡化的其他视觉元素当作背景。其实，这就是我们今天要讲的第一个方法，建立视觉层次。

我们以上一节课平均温度与露点的散点图为例，来说说这具体是怎么做。



我们知道，一个可视化图表应该包括图形、图例、提示信息、坐标轴等等元素。其中，图形是最重要的元素，所以它一般会在图表最核心的区域呈现。上图中，我们就使用了比较鲜明的蓝色来突出图形。至于左侧和下方的坐标轴，我们就用比较淡的灰黑色来显示。背景中的辅助线存在感最弱，因为它们是用来辅助用户更认真地阅读图表、理解数值的，不

是主要元素，所以我们会用非常淡的颜色把它们显示在背景里。这些元素就构成了一个有鲜明视觉层次感的图表。

不过，就这个图表而言，我们还可以把它做得更好。因为，我们实际上希望表达给用户的信息还包含了平均气温与露点的正相关性，如果用户对这个数学关系比较敏感，完全可以通过散点分布了解到它们的正相关性，但是对其他不敏感的用户来说，我们可以添加曲线图来引导他们。接下来，我们一起来看具体的做法，从中你就可以体会到建立视觉层次思路了。

第一步，我们处理一下数值，将数据按照气温高低排序，然后对相同温度的数据进行分组，最后将相同温度下的平均露点温度计算出来。

[复制代码](#)

```
1 // 露点排序
2 let dataset2 = [...dataset].sort((a, b) => a.tdp - b.tdp);
3
4 // 对相同露点的温度进行分组
5 dataset2 = dataset2.reduce((a, b) => {
6   let curr = a[a.length - 1]
7   if (curr && curr.tdp === b.tdp) {
8     curr.temperature.push(b.temperature)
9   } else {
10    a.push({
11      temperature: [b.temperature],
12      tdp: b.tdp
13    })
14   }
15   return a
16 }, []);
17
18 // 最后将露点平均温度计算出来
19 dataset2 = dataset2.map(d => {
20   d.category = '露点平均气温'
21   d.temperature = Math.round(d.temperature.reduce((a, b) => a + b) / d.tempera
22   return d
23 })
```

在计算好数据之后，我们将散点和曲线两个数组都传给图表对象。

[复制代码](#)

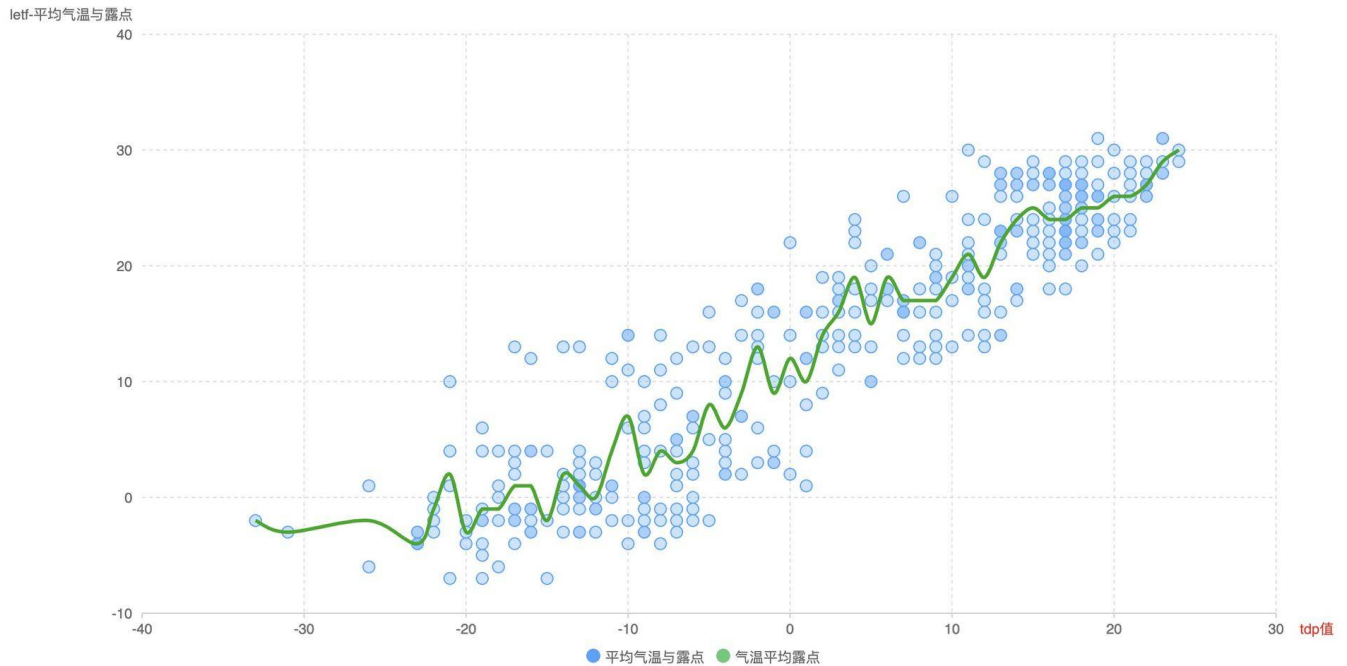
```
1 const chart = new Chart({
2   container: '#app'
```

```
3 });  
4 let clientRect={bottom:50};  
5 chart.source([...dataset, ...dataset2], {  
6   row: 'category',  
7   value: 'temperature',  
8   text: 'tdp'  
9 });
```

最后，我们就能分别用散点和曲线图来呈现数据了

复制代码

```
1 const ds = chart.dataset;  
2 const d1 = ds.selectRows("平均气温与露点");  
3 const d2 = ds.selectRows('露点平均气温');  
4  
5 // 散点图  
6 const scatter = new Scatter({  
7   clientRect,  
8   showGuideLine: true,  
9 }).source(d1);  
10  
11 // 曲线图  
12 const line = new Line().source(d2);  
13 line.style('line', function(attrs, data, i) {  
14   return { smooth: true, lineWidth: 3, strokeColor: '#0a0' };  
15 });  
16 line.style('point', function(attrs) {  
17   return { display: 'none' };  
18 });  
19
```

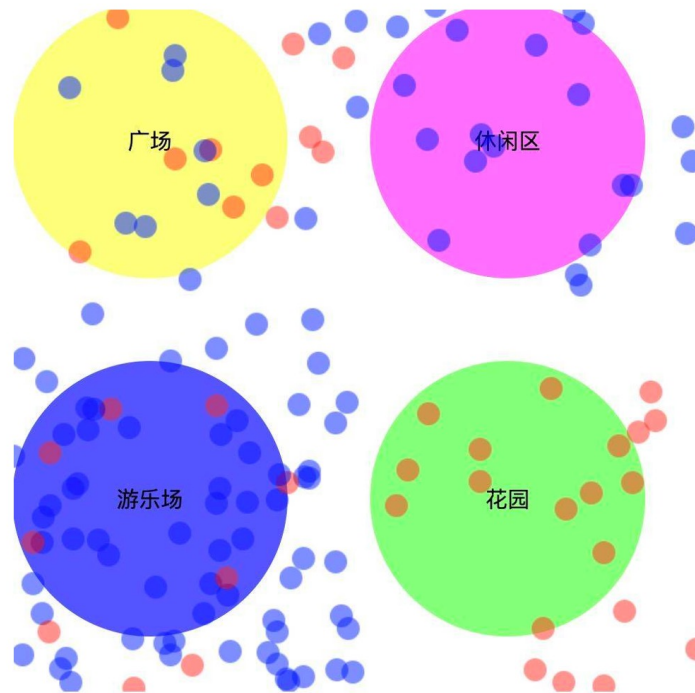


这个图表分为三个层次：第一层，我们用曲线描绘气温与平均露点的关系，随着气温升高，平均露点温度也会升高，二者的总体趋势保持一致。第二层，就是我们保留的散点图，我们可以通过它看出具体某一次记录的温度和露点数据，还可以从分布规律看出相关性的强度，因此它能够表达的信息比曲线图还要多一些。第三层，我们使用了坐标轴和辅助线作为背景。总之，像这样层次分明的图表，非常有助于我们快速理解图表上的信息。

## 选择合适图表，直观表达信息

建立视觉层次是第一种集中注意力的方法，第二种方法其实和它有点类似，就是用合适的图表更加直观地去表达信息。

还记得我们在 32 节课中讲的公园游客散点图吗？我们将男游客和女游客分别标记之后，可以看到不同区域在某个时刻的游客性别分布，比如下图就是公园 12 点游客分布情况。



公园12点游客散点图

不过，散点图虽然能够一眼看出不同性别游客在四个区域的大致分布，但不够直观。如果我们要更精细地分析的话，还是应该对数据呈现方式进行改进。

在表达某个单组变量的分布状况的时候，使用饼图是一个比较直观的方式。

比如，我们可以用一组饼图来表示中午 12 点公园游客在四个区域的性别分布，让它对应上面那张散点图。

第一步是处理数据。我们就以给 12 点的游客分类为例。我们根据游客所在的区域和性别来统计人数，对应到下面的代码，就是 countPeople 函数中实现的逻辑。

[复制代码](#)

```
1 function count(d, dataset) {  
2   let place;  
3   if(d.x < 300 && d.y < 300) {  
4     place = 'square';  
5   } else if (d.x >= 300 && d.y < 300) {  
6     place = 'sections';  
7   } else if (d.x >= 300 && d.y >= 300) {  
8     place = 'garden';  
9   } else {  
10    place = 'playground';  
11  }  
12  dataset[place] = dataset[place] || []
```

```
13     {
14         gender: '男游客',
15         people: 0,
16     },
17     {
18         gender: '女游客',
19         people: 0,
20     }
21 ];
22 if(d.gender === 'f') {
23     dataset[place][0].people++;
24 } else {
25     dataset[place][1].people++;
26 }
27 return dataset;
28 }
29
30 function groupData(data) {
31     const dataset = {};
32
33     for(let i = 0; i < data.length; i++) {
34         const d = data[i];
35
36         if(d.time === 12) {
37             const p = count(d, dataset);
38         }
39     }
40     return dataset;
41 }
42
43 const dataset = groupData(data);
```

经过这么分类之后，现在的数据集看起来就是下面这样：



```

▼ {garden: Array(2), playground: Array(2), sections: Array(2), square: Array(2)} ⓘ
  ▼ garden: Array(2)
    ▶ 0: {gender: "男游客", people: 20}
    ▶ 1: {gender: "女游客", people: 0}
    length: 2
    ▶ __proto__: Array(0)
  ▼ playground: Array(2)
    ▶ 0: {gender: "男游客", people: 9}
    ▶ 1: {gender: "女游客", people: 65}
    length: 2
    ▶ __proto__: Array(0)
  ▼ sections: Array(2)
    ▶ 0: {gender: "男游客", people: 0}
    ▶ 1: {gender: "女游客", people: 23}
    length: 2
    ▶ __proto__: Array(0)
  ▼ square: Array(2)
    ▶ 0: {gender: "男游客", people: 11}
    ▶ 1: {gender: "女游客", people: 13}
    length: 2
    ▶ __proto__: Array(0)
  ▶ __proto__: Object

```

第二步，我们用 qcharts 来创建四个饼图。在图中，我们用橙色表示女游客，绿色表示男游客。这样，我们就能明显看出四个区域中男、女游客的分布情况了。具体的代码如下：

[复制代码](#)

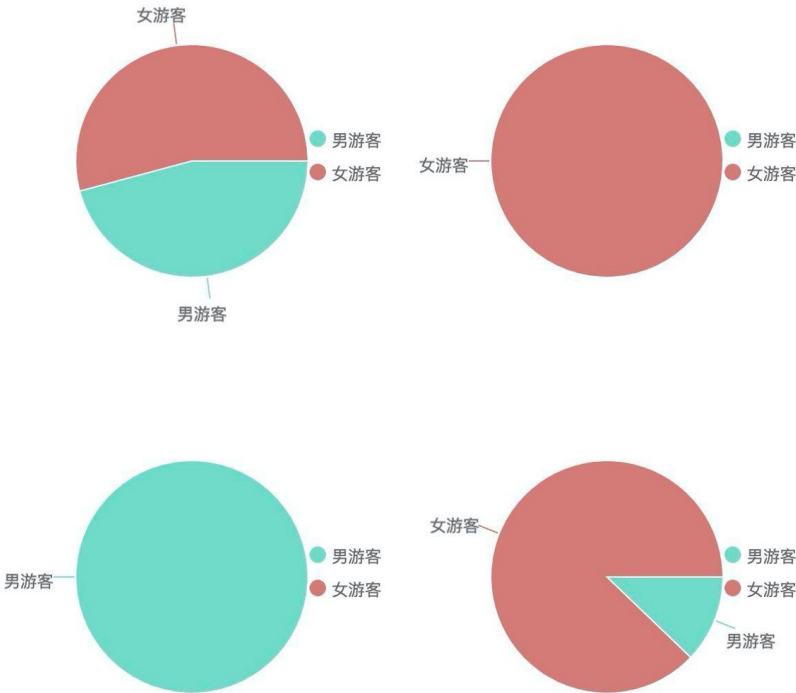
```

1  const { Chart, Pie, Legend, Tooltip, theme } = qcharts;
2  theme.set({
3    colors: ['#71dac7', '#d57a77'],
4  });
5
6  Object.entries(dataset).forEach(([key, dataset]) => {
7    const chart = new Chart({
8      container: `#${key}`
9    });
10   chart.source(dataset, {
11     row: 'gender',
12     value: 'people',
13     text: 'gender'
14   });
15   const pie = new Pie({
16     radius: 0.7,
17     animation: {
18       duration: 700,
19       easing: 'bounceOut'
20     }
21   });
22   const legend = new Legend({ orient: 'vertical', align: ['right', 'center'] });
23   const toolTip = new Tooltip();
24   chart.append([pie, legend, toolTip]);

```

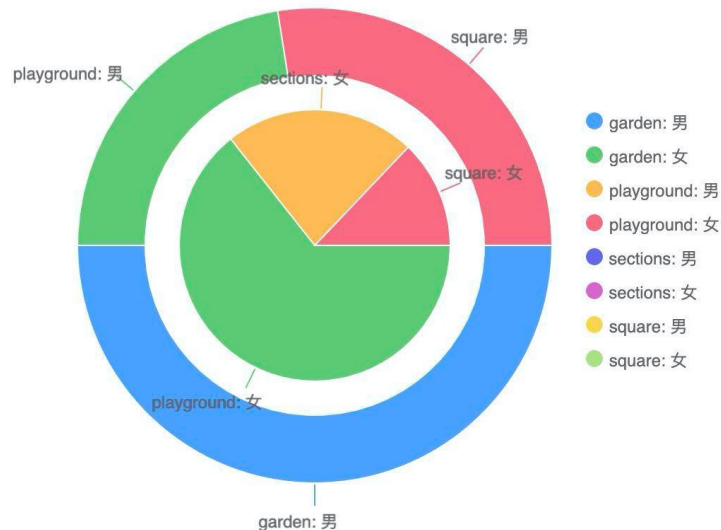


25 } ) :



虽然饼图表示的结果非常简单和直观，但它也有缺点，就是一张饼图实际上能表示的信息很少。一个饼图一次只能表示一组单维度数据，而性别又只有男和女两种，所以我们为了表示四个区域，不得不用四张饼图，这会非常麻烦。

那我们可以尝试用一张图表来表示更多维度的信息吗？这当然可以。我们可以尝试把前面的四张饼图合并成一张**嵌套饼图**，它由两个饼状图组成，中间小的饼状图是女性在四个区域的分布情况，大的饼图是男性在四个区域的分布情况。



它的具体实现方法是，首先，我们在前面 `groupData` 的基础上对数据进行进一步处理，将数据对象扁平化，然后添加 `place` 属性。

复制代码

```
1 const dataset = [];
2 Object.entries(groupData(data)).forEach(([place, d]) => {
3   d[0].place = `${place}: 男`;
4   d[1].place = `${place}: 女`;
5   dataset.push(...d);
6 });
```

这样处理数据之后，新的数据结构如下：

```
▼ (8) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}] ⓘ
▶ 0: {gender: "男游客", people: 20, place: "garden: 男"}
▶ 1: {gender: "女游客", people: 0, place: "garden: 女"}
▶ 2: {gender: "男游客", people: 9, place: "playground: 男"}
▶ 3: {gender: "女游客", people: 65, place: "playground: 女"}
▶ 4: {gender: "男游客", people: 0, place: "sections: 男"}
▶ 5: {gender: "女游客", people: 23, place: "sections: 女"}
▶ 6: {gender: "男游客", people: 11, place: "square: 男"}
▶ 7: {gender: "女游客", people: 13, place: "square: 女"}
```

charts3.html:79

然后，我们可以在这个数据结构的基础上绘制两个饼图，代码如下：


```
1  const { Chart, Pie, Legend, Tooltip, theme } = qcharts;
2  const chart = new Chart({
3    container: `#container`
4  });
5
6  chart.source(dataset, {
7    row: 'place',
8    value: 'people'
9  });
10 const ds = chart.dataset;
11
12 const pie = new Pie({
13   radius: 0.4,
14   pos: [0, 0]
15 }).source(ds.selectRows(dataset.filter(d => d.gender === '女游客')).map(d => d.p
16
17 const pie2 = new Pie({
18   innerRadius: 0.5,
19   radius: 0.7
20 }).source(ds.selectRows(dataset.filter(d => d.gender === '男游客')).map(d => d.p
21
22 const legend = new Legend({ orient: 'vertical', align: ['right', 'center'] });
23
24 chart.append([pie2, pie, legend]);
```

这样我们就用一张图表直观地显示了，中午 12 点，公园内四个区域中男女游客的分布情况了。不过实际上，这张图表和前面的四个饼图表达的信息不同，那四张饼图表达的是某个区域男女游客分布的情况，而这张图则是表达男女游客分别在四个区域的分布情况，它们正好是互补的。

那么有没有办法将游客性别和游客区域的分布情况融合在一起表达呢？这也是可以的。我们可以用堆叠的直方图，或者更加美观的南丁格尔玫瑰图来表达。这里，我选择用南丁格尔玫瑰图。


南丁格尔玫瑰图是一种圆形的直方图，用半径来表示数量和数据之间的区别。在这一张图上我们可以看出，四个区域的总人数分布，以及每个区域男女游客数量分布。

与上面的嵌套饼图一样，我们也要先把数据扁平化，不过这里要稍微修改了一下 place 属性。因为我们要将数据堆叠，所以让男女游客数据里，在同一个区域的游客统计的 place 属性保持一致就行了。

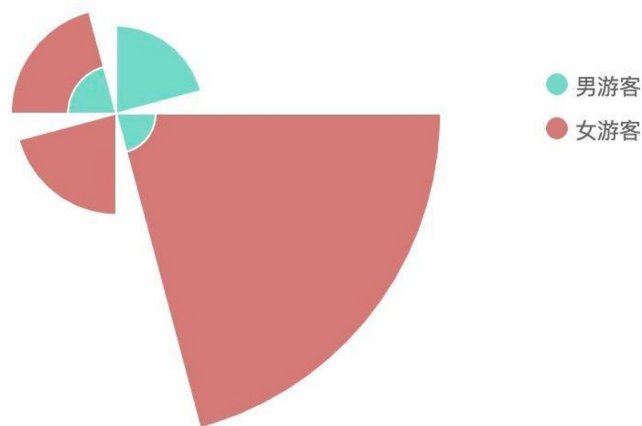
 复制代码

```
1  const dataset = [];  
2  Object.entries(groupData(data)).forEach(([place, d]) => {  
3    d[0].place = place;  
4    d[1].place = place;  
5    dataset.push(...d);  
6  });
```

明白了原理，我们就可以绘制出南丁格尔玫瑰图了。

 复制代码

```
1  const { Chart, PolarBar, Legend, Tooltip, theme } = qcharts;  
2  const chart = new Chart({  
3    container: `#container`  
4  });  
5  
6  theme.set({  
7    colors: ['#71dac7', '#d57a77'],  
8  });  
9  
10 chart.source(dataset, {  
11   row: 'gender',  
12   value: 'people',  
13   text: 'place',  
14 });  
15 const bar = new PolarBar({  
16   stack: true,  
17   radius: 0.8,  
18   groupPadAngle: 15,  
19 }).style("pillar", {  
20   strokeColor: "#FFF",  
21   lineWidth: 1,  
22 });  
23 const tooltip = new Tooltip();  
24 const legend = new Legend({ orient: 'vertical', align: ['right', 'center'] });  
25  
26 chart.append([bar, tooltip, legend]);
```

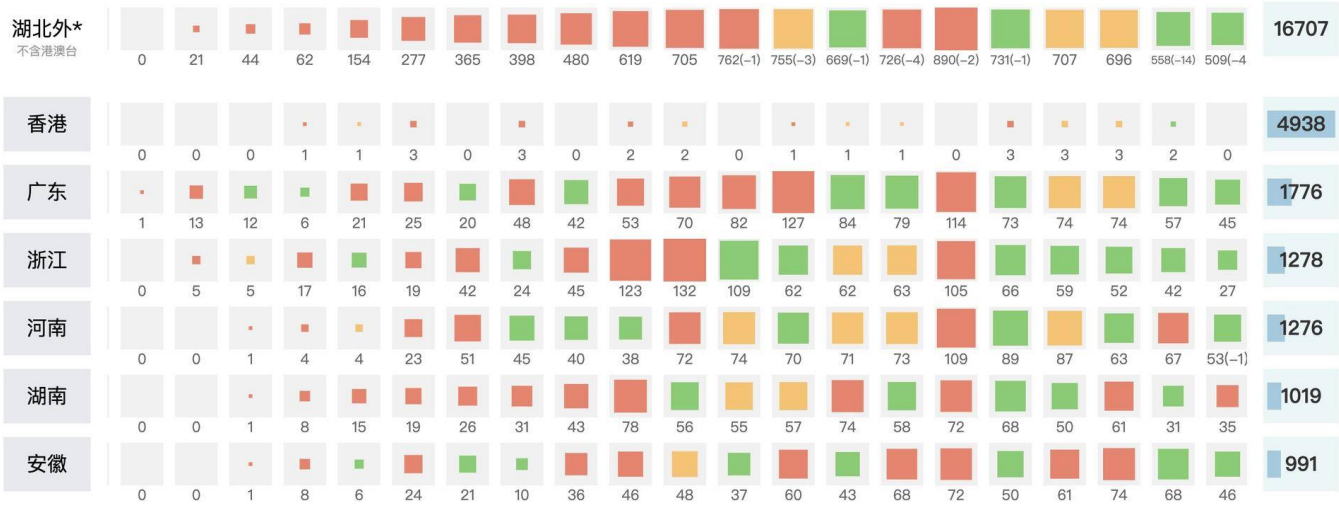


具体的方法讲完了，我们做个总结。使用南丁格尔玫瑰图，我们能把人群在公园区域的分布和性别分布规律显示在一张图上，让更多的信息呈现在一张图表上。这既能节省空间，也便于人们高效率地获取更多信息。但是，太多的信息聚集也会显著增加图表复杂度，减少图表的直观程度。就像这张南丁格尔图一样，它虽然简单，但直观性仍然不如之前用四个饼图和一个嵌套饼图的表达形式。所以在我们实际可视化项目中，需要根据实际情况选择合适的解决方案，大部分情况下，我们需要在直观性和信息聚集程度上做一个取舍。

## 改变图形属性，强化数据差异

除了直观表达信息外，我们还可以采用一些其他的手段，比如，改变颜色、大小、形状等等，以此来强化数据之间的差异，这也是增强可视化图表中信息表达的一种手段。

这次我们就不具体实现了，直接来看一个成熟的例子。比如说，北大可视化实验室设计的全国新冠病毒肺炎疫情晴雨表，就是使用颜色和方块大小，将增量数据很直观地表达出来，从而宏观地呈现出疫情的发展态势，这对于揭示疫情拐点来说非常有帮助。



颜色和面积、折线图的方向、直方图的高度差，这些方式都能比较鲜明地体现出数据之间差异。除此之外，我们在绘制图表的时候，还可以使用背景网格线，来辅助用户观察数据间的差异，发现数据之间的变化规律。

说到强调数据差异，我还想给你说一种比较完美的图表，它就是股市中常用的蜡烛图，又叫做 **K 线图**。



我要先说明一点，我不鼓励任何人进行投机性的炒股。但不得不说，股市里的 K 线图是一个非常成功的可视化案例，这个图表用颜色来强化数据的增减，还包含了许多其他有用的信息，让想要了解股市市场的人，能够从中分析出商品或者股票的价格走势，再做出相应的决策。



总之，从这些优秀的案例中我们知道，在可视化实现中，我们应该重视数据之间的比较，用一些图形属性来强调数据的差异，这对加强图表的表现力非常有效。

## 要点总结

这节课，我们学习了让可视化设计更加清晰的三种方法。首先，我们应该学会建立视觉层次，信息有主次之分，我们要把重要的信息突出显示，减少次要信息对比，以及干扰信息的存在感。

其次，我们要学会用合适的图表来直观地表达信息，比如说，一般情况下，我们会用饼图来描述变量值的分布，用嵌套饼图来展现更多信息，用南丁格尔玫瑰图聚合多维度信息。

最后，我们还应该重视数据之间的比较。大部分情况下，我们会使用一些图形属性，比如更改图形颜色、形状等等，来强调数据之间的差异，这样能够增强图表的表现力。

总的来说，可视化设计的许多方法，其实都是源于实践经验的积累。想要学好，真的没有捷径可走，今天我讲的这三种方法也都是特例。学会它们之后，你还是要多练习，争取做到举一反三，这样才能在可视化设计这条路上走得更远。

## 小试牛刀

你可以利用我放在 [🔗 GitHub](#) 仓库里的天气和空气质量数据，设计出可视化案例，展现出和散点图不一样的内容吗？比如说，你可以让图表展现不同月份下，晴天和雾霾天的分布率。

当然，如果你有更好的创意，我非常鼓励你把它们分享出来，期待在留言区看到你的作品。今天就讲到这里，我们下节课见！

---

## 源码

🔗 示例代码详见 [GitHub 仓库](#)



提建议

更多课程推荐

# 数据结构与算法之美

为工程师量身打造的数据结构与算法私教课

王争

前 Google 工程师



立省 ¥40

破 90000 订阅特惠，到手价 ¥89

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 34 | 数据处理（二）：如何处理多元变量？

下一篇 36 | 设计（二）：如何理解可视化设计原则？

## 精选留言

写留言

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。