Squarified Treemap 算法作业报告

张启哲　1900011638

2021 年 11 月 8 日

目录 (Contents)

1 数据描述与分析 (Data Description and Analysis)

1.1 数据描述

本次可视化作业使用的数据是来自于China Biographical Database Project(CBDB)中 Ming Jinshi List 的明进士数据。数据集包括明代 52 年科举考试的所有进士原始资料，共 14116 人。

数据集以 excel 数据 xlsx 的格式呈现，共有 3 个工作表。第 1 个工作表 Ming Jinshi Lists 中包括了所有进士的详细信息，第 2 个工作表 Release Notes 中描述了 3 个工作表中的内容并列出了各次科举考试的年份和资料出处，第 3 个工作表 dazi rule 描述了进士数据项的格式。

1.2 数据分析

本次可视化算法 Squarified Treemap 处理的是层次数据，根据首个工作表中的各项目属性，选择籍贯一项进行处理。明代的行政区划大致可分为司府州县 4 级与司州县、司府州 3 级两种制度并存，在本次作业中主要将籍贯按照级别划分为 3 类：司、府 (州)、州 (县)，府与州合并，将直隶州与府视为同一级别处理，州与县合并，县归属于非直隶的州管理，无归属的则视为与州同一级别的独立行政单位。此外，明代还设有专门的军籍制度，下设卫、所，本次作业将卫所合并视为与司同一级的行政单位，而分别视为与府同一级的行政单位，以下不分 (许多带有地域的卫所的治所并不在当地，因此将其划为当地也不妥)。

1.3 数据清洗

本次作业在清洗数据时首先将 xlsx 格式的数据转化为 json 格式的数据，之后根据各级行政单位逐级拆分，部分含有罕用字的籍贯在安装了 Unicode extension A & B 后依然无法读取，这部分籍贯被划为未知 (数量较少)，同时有极少数特殊籍贯 (如高丽等) 也做了同样处理。

值得一提的是，原始数据中的许多籍贯在录入时没有记录上一级行政单位，为了形成层次结构数据，处理时首先将数据集扫描一遍，记录其中已知的行政区划从属关系，在第二遍处理数据时便可以将大多数无上级行政单位的籍贯正确划分 (例如第 1 位进士的籍贯是 X 府 Y 州，第 2 位的是 Y 州，此时原始数据未记录第 2 位进士所属的府，但可以通过第 2 位进士的籍贯将第 2 位正确划分为 X 府)，剩余的小部分未知归属划为已知最低一级行政单位下的其他分支。

另外，在较长的时间推移中，不少地名出现了演变，如宁波又名明州、宁江、顺天又名北平等，不少地名中出现异体字 (即同一个字有不同的写法)，如“凉”与“涼”、“寧”与“甯”等，还有原始数据在录入时可能存在错别字的现象，如将莱州误记为菜州、绍兴误记为照兴等，本次作业在清洗数据时均已充分考虑，并最大限度地保证清洗过后数据的真实性 (具体细节在数据处理文件中可见)。同时，原始数据在南直隶与北直隶上未作区分，统一记为直隶，但考虑到南北直隶在实际地理位置上有较大差别，本次作业将这两个直隶司在数据清洗时进行了拆分。

2 算法设计与分析 (Algorithm Design and Analysis)

本次作业中的算法直接复现了 Mark Bruls, Kees Huizing, and Jarke J. van Wijk 的论文Squarified Treemaps中的算法 (算法实现细节在 js 文件中可见)。

其中主算法 squarify 通过递归的方式实现:

Algorithm 1 squarify

Input: children: list of child nodes; row: list of nodes in current row; w: width of current row.

```
1: if children == [] then
2:     if row != [] then
3:         layoutrow(row, w);
4:     end if
5:     return;
6: end if
7: var node = children[0];
8: if worst(row, w) >= worst(row++[node], w) then
9:     squarify(row--[node], row++[node], w);
10: else
11:     layoutrow(row, w);
12:     squarify(children, [], min(rwidth, rheight));
13: end if
```

其中 [] 代表空列表，++ 和 --分别代表列表元素的增添和删除，layoutrow 只是简单地记录已经确定并摆放好位置的一行矩形，worst 则是计算一行矩形中的最大长宽比，实现如下:

Algorithm 2 worst

Input: row: list of nodes in current row; w: width of current row.

Output: ratio: highest aspect ratio of current row.

```
1: var s = sum(row);
2: var r+ = max(row);
3: var r- = min(row);
4: ratio = max((w^2 r+)/(s^2), (s^2)/(w^2 r-));
```

而 rwidth 与 rheight 则是指剩余矩形的宽与高 (这里省去了原算法中的 width() 函数)。

注：在绘制树图时引入 parent 属性来分配颜色，一级子结点的 parent 设置为自己，其余子结点的 parent 都递归地设置为父结点，使得颜色按照第一级行政区划 (司) 分配；每个小矩形的颜色透明度与字号大小都与面积正相关，以使整个树图更加直观与美观。同时可视化实现中加入了用户交互，上方按钮条可以选择展示的行政区划级别，用户将鼠标放置在小矩形上时对应矩形会高亮显示，鼠标点击时会弹出该矩形对应的行政区划名称与进士总人数。

3 可视化结果描述 (Visualization Result Description)

3.1 结果展示

代码库已上传至 github: Theia-4869/Squarified-Treemap。

Figure 1: Squarified Treemap of Si-level

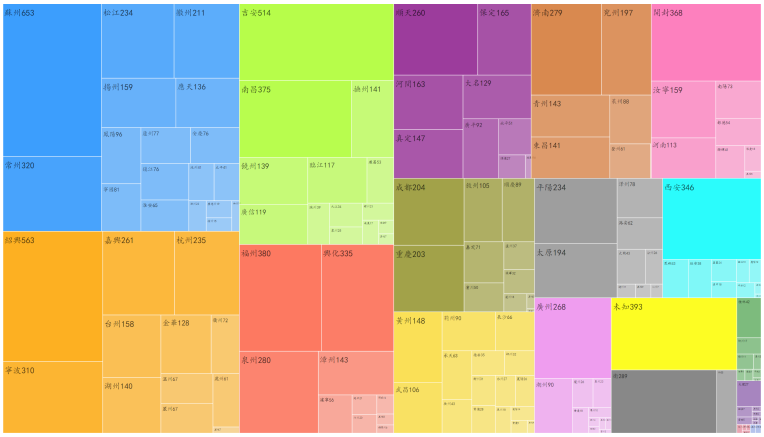

Figure 2: Squarified Treemap of Fu-level

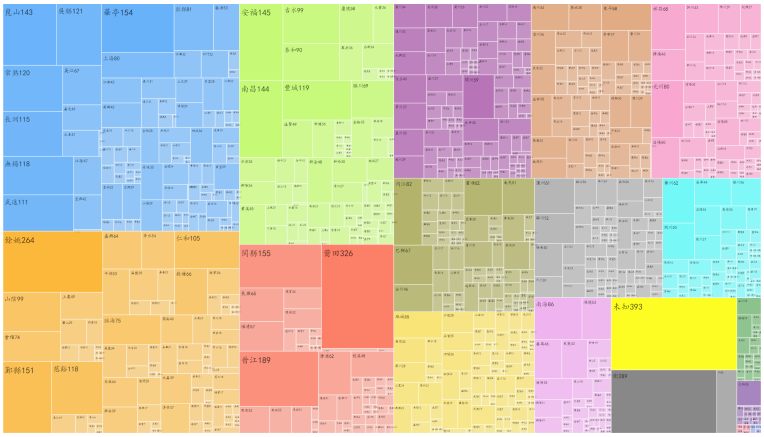

Figure 3: Squarified Treemap of Xian-level

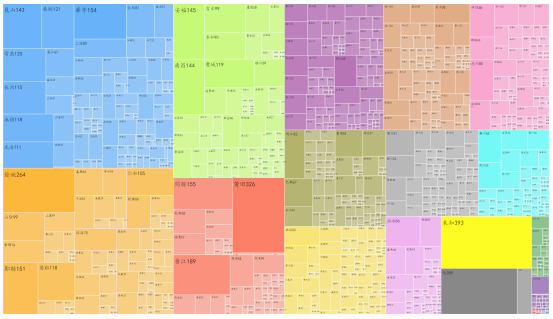

Figure 4: Squarified Treemap

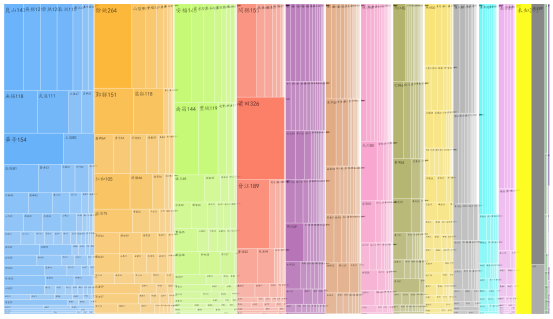

Figure 5: Simple Treemap

3.2 总结与分析

从数据处理过程中发现，原始数据集存在诸多问题，如归类不清、错误记录等，但可利用一些手段最大限度地恢复数据中的真实信息，因此数据处理在可视化的任务中至关重要，往往也消耗了最多的时间 (本次作业就是这样)。但好的数据处理也能够表现出巨大的功效，如在此次处理明进士籍贯的任务中，如果不加区分只是单纯简单地按照录入顺序清洗数据，划入未知类的数据可能达到两、三千条，但在精细地筛查后可将这一数字缩小至三、四百，减小了约一个数量级，能够为可视化提供更加准确的信息。

从可视化结果中发现，历代考取进士最多的司级行政单位是南直隶 (2333)，最少的是贵州 (4)；最多的府级行政单位是苏州 (653)；最多的州县级行政单位则是莆田 (326)。总体上各区域考取进士的人数与行政区划管辖面积及经济发展情况都正相关。南北直隶司下辖面积较大，考取进士人数相应多，而经济高度发展的江南地区 (南直隶、浙江、江西) 进士数量也均位居前列，较为偏远的交趾、贵州等地区教育程度则相对落后，考取进士零星 (其中也有边远地区户籍管理不规范导致划分遗漏的原因)。大部分地区的进士考取都相对分散，只有福建等少数地区较为集中，这反映出当时中国的教育资源至少在司辖域内还是分配较为均衡的。同时，注意到两件有趣的事，其一是明代才子文武双全，在考中进士的 14116 人中便有 344 人可能出自军籍，与一个小型的司级行政区考取的人数相当；其二则是原始数据集中也出现了明代有记载的唯一一位外籍进士——高丽国金涛，他在洪武四年考中辛亥科三甲第五名进士，被授任县丞，而后本人以不通汉语为理由，请求回国，可谓是相当有意思了。

从算法设计及最终实现过程中发现，Squarified Treemap 算法绘制出的等方树图与简单树图相比小矩形的长宽比更趋向 1，而正方形更易于观察与点击，故等方树图的可视化效果显然更优。而算法绘制出的等方树图的美观程度与小矩形面积及原始矩形的给定长宽关系紧密，绘制时应当保证小矩形面积之和等于原始矩形的面积 (不相等时可将小矩形面积按比例扩大)，否则将会留下大块空白区域，同时原始矩形的长宽比也至关重要，过大的长宽比同样会留下大片空白，而长宽趋向相等时很难与显示器尺寸适配，只有一个较小范围内的值可以适配当前数据，在绘制时需要反复调整。