3022207128-杨宇鑫-实验报告 4-3

1. 实验要求:

实验 3： 绘制世界 GDP 和人口分布

使用 d3 Voronoi 分别绘制世界各国 GDP 和人口分布。数据：countries.csv，按照地区划分为两层。

要求：效果参考下图。将 2008 到 2106 的数据累加起来。

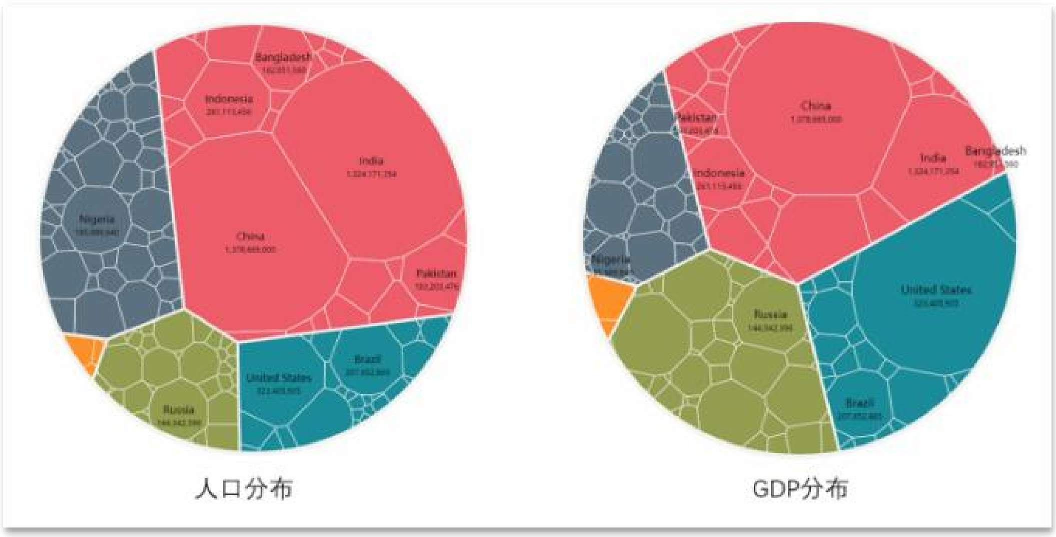


图 24 实验 3 效果参考图

2. 实验过程:

2.1 处理数据

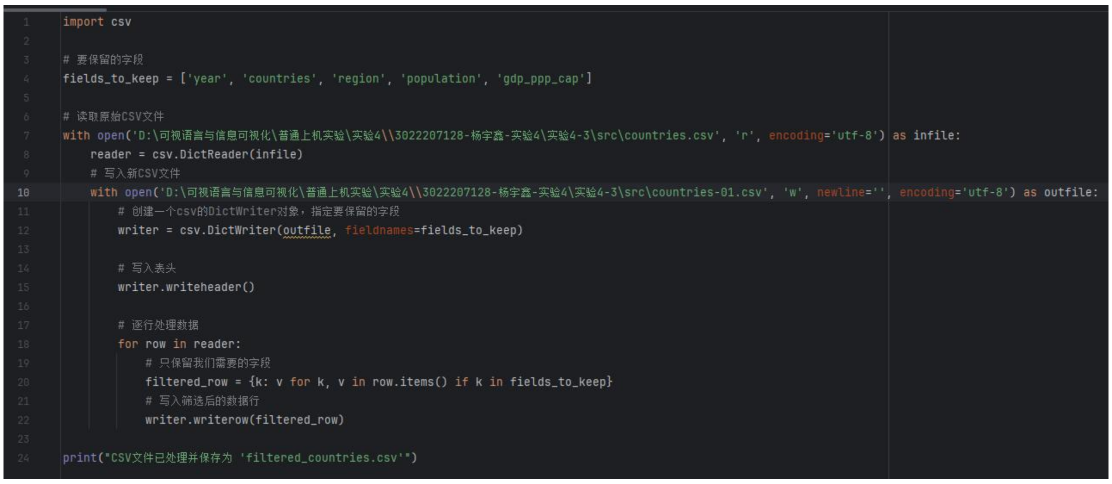
2.1.1 编写 python 脚本处理数据

2.1.1.1 下面这么多的字段只保留 year,countries,region,population,gdp_ppp_cap 字段

```
year,ISO_code,countries,region,pf_rol_procedural,pf_rol_civil,pf_rol_criminal,pf_rol,pf_ss_homicide,pf_ss_disappearances_disap,pf_ss_disappearances_violent,pf_ss_disappearances_organized,pf_ss_disappearances_fatalities,pf_ss_disappearances_injuries,pf_ss_disappearances,pf_ss_women_fgm,pf_ss_women_missing,pf_ss_women_inheritance_widows,pf_ss_women_inheritance_daughters,pf_ss_women_inheritance,pf_ss_women,pf_ss,pf_movement_domestic,pf_movement_foreign,pf_movement_women,pf_movement,pf_religion_estop_establish,pf_religion_estop_operate,pf_religion_estop,pf_religion_harassment,pf_religion_restrictions,pf_religion,pf_association_association,pf_association_assembly,pf_association_political_establish,pf_association_political_operate,pf_association_political,pf_association_prof_establish,pf_association_prof_operate,pf_association_prof,pf_association_sport_establish,pf_association_sport_operate,pf
```

```
_association_sport,pf_association,pf_expression_killed,pf_expression_jailed,
pf_expression_influence,pf_expression_control,pf_expression_cable,pf_expres
sion_newspapers,pf_expression_internet,pf_expression,pf_identity_legal,pf_i
dentity_parental_marriage,pf_identity_parental_divorce,pf_identity_parental,
pf_identity_sex_male,pf_identity_sex_female,pf_identity_sex,pf_identity_div
orce,pf_identity,pf_score,pf_rank,ef_government_consumption,ef_government_t
ransfers,ef_government_enterprises,ef_government_tax_income,ef_government_t
ax_payroll,ef_government_tax,ef_government,ef_legal_judicial,ef_legal_court
s,ef_legal_protection,ef_legal_military,ef_legal_integrity,ef_legal_enforce
ment,ef_legal_restrictions,ef_legal_police,ef_legal_crime,ef_legal_gender,e
f_legal,ef_money_growth,ef_money_sd,ef_money_inflation,ef_money_currency,ef
_money,ef_trade_tariffs_revenue,ef_trade_tariffs_mean,ef_trade_tariffs_sd,e
f_trade_tariffs,ef_trade_regulatory_nontariff,ef_trade_regulatory_complianc
e,ef_trade_regulatory,ef_trade_black,ef_trade_movement_foreign,ef_trade_mov
ement_capital,ef_trade_movement_visit,ef_trade_movement,ef_trade,ef_regulat
ion_credit_ownership,ef_regulation_credit_private,ef_regulation_credit_inte
rest,ef_regulation_credit,ef_regulation_labor_minwage,ef_regulation_labor_f
iring,ef_regulation_labor_bargain,ef_regulation_labor_hours,ef_regulation_l
abor_dismissal,ef_regulation_labor_conscription,ef_regulation_labor,ef_regu
lation_business_adm,ef_regulation_business_bureaucracy,ef_regulation_busine
ss_start,ef_regulation_business_bribes,ef_regulation_business_licensing,ef_
regulation_business_compliance,ef_regulation_business,ef_regulation,ef_scor
e,ef_rank,hf_score,hf_rank,hf_quartile,population,gdp_ppp_cap,region_simple,
pop_rank,gdp_rank
```

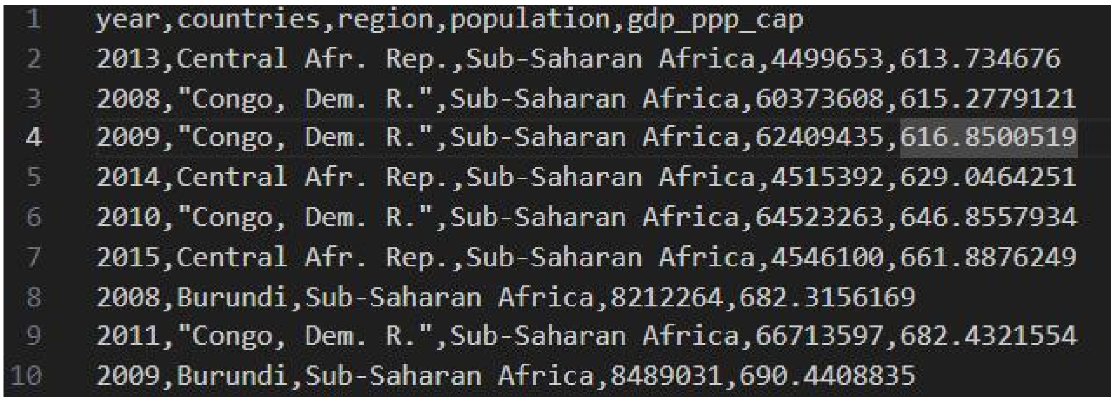
python 处理脚本:



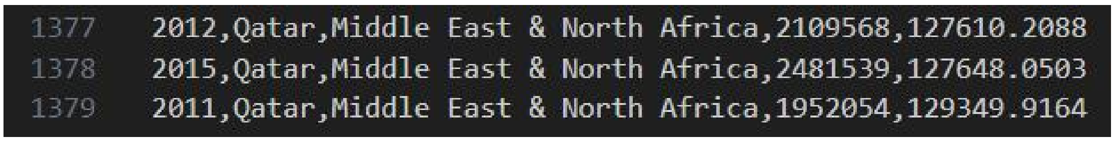
```python
import csv

# 要保留的字段
fields_to_keep = ['year', 'countries', 'region', 'population', 'gdp_ppp_cap']

# 读取原始CSV文件
with open('D:\可视语言与信息可视化\普通上机实验\实验4\\3022207128-杨宇鑫-实验4\实验4-3\src\countries.csv', 'r', encoding='utf-8') as infile:
    reader = csv.DictReader(infile)
    # 写入新CSV文件
    with open('D:\可视语言与信息可视化\普通上机实验\实验4\\3022207128-杨宇鑫-实验4\实验4-3\src\countries-01.csv', 'w', newline='', encoding='utf-8') as outfile:
        # 创建一个csv的DictWriter对象，指定要保留的字段
        writer = csv.DictWriter(outfile, fieldnames=fields_to_keep)

        # 写入表头
        writer.writeheader()

        # 逐行处理数据
        for row in reader:
            # 只保留我们需要的字段
            filtered_row = {k: v for k, v in row.items() if k in fields_to_keep}
            # 写入筛选后的数据行
            writer.writerow(filtered_row)

print("CSV文件已处理并保存为 'filtered_countries.csv'")
```

```python
import csv
# 要保留的字段
fields_to_keep = ['year', 'countries', 'region', 'population', 'gdp_ppp_cap']
# 读取原始 CSV 文件
with open('D:\可视语言与信息可视化\普通上机实验\实验 4\\3022207128-杨宇鑫-实验 4\
实验 4-3\src\countries.csv', 'r', encoding='utf-8') as infile:
    reader = csv.DictReader(infile)
    # 写入新 CSV 文件
    with open('D:\可视语言与信息可视化\普通上机实验\实验 4\\3022207128-杨宇鑫-实验 4\实验 4-3\src\countries-01.csv', 'w', newline='', encoding='utf-8') as outfile:
        # 创建一个 csv 的 DictWriter 对象，指定要保留的字段
        writer = csv.DictWriter(outfile, fieldnames=fields_to_keep)
        # 写入表头
        writer.writeheader()
        # 逐行处理数据
        for row in reader:
            # 只保留我们需要的字段
            filtered_row = {k: v for k, v in row.items() if k in fields_to_keep}
            # 写入筛选后的数据行
            writer.writerow(filtered_row)
print("CSV 文件已处理并保存为 'filtered_countries.csv'")
```

处理结果:



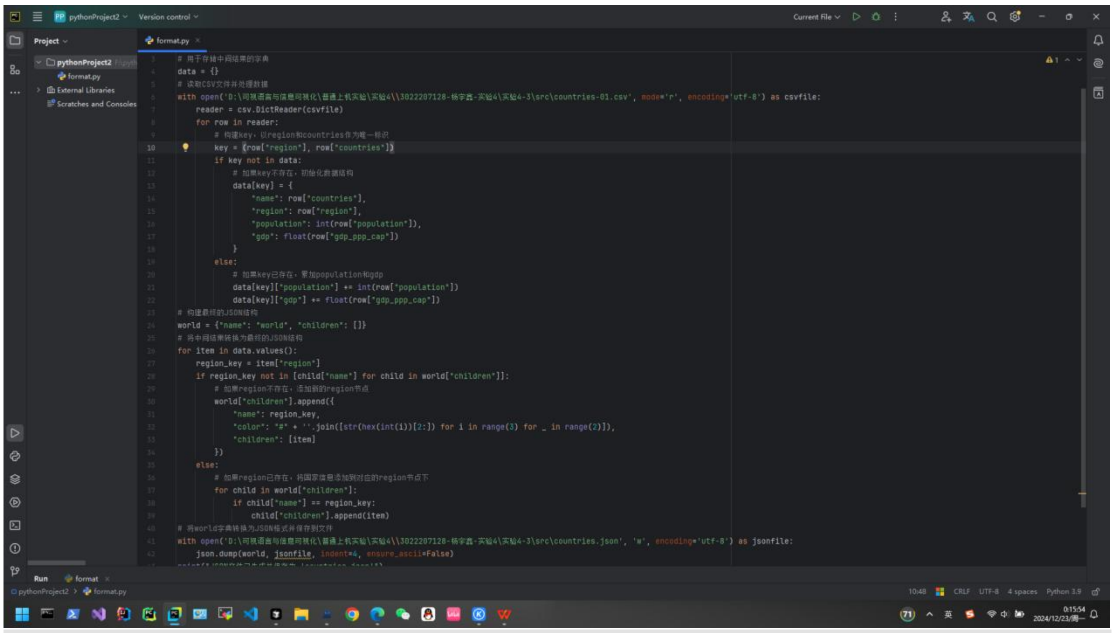
```
year,countries,region,population,gdp_ppp_cap
2013,Central Afr. Rep.,Sub-Saharan Africa,4499653,613.734676
2008,"Congo, Dem. R.",Sub-Saharan Africa,60373608,615.2779121
2009,"Congo, Dem. R.",Sub-Saharan Africa,62409435,616.8500519
2014,Central Afr. Rep.,Sub-Saharan Africa,4515392,629.0464251
2010,"Congo, Dem. R.",Sub-Saharan Africa,64523263,646.8557934
2015,Central Afr. Rep.,Sub-Saharan Africa,4546100,661.8876249
2008,Burundi,Sub-Saharan Africa,8212264,682.3156169
2011,"Congo, Dem. R.",Sub-Saharan Africa,66713597,682.4321554
2009,Burundi,Sub-Saharan Africa,8489031,690.4408835
```



```
2012,Qatar,Middle East & North Africa,2109568,127610.2088
2015,Qatar,Middle East & North Africa,2481539,127648.0503
2011,Qatar,Middle East & North Africa,1952054,129349.9164
```

2.1.1.2 将 csv 格式文件用 python 脚本处理成 json 文件

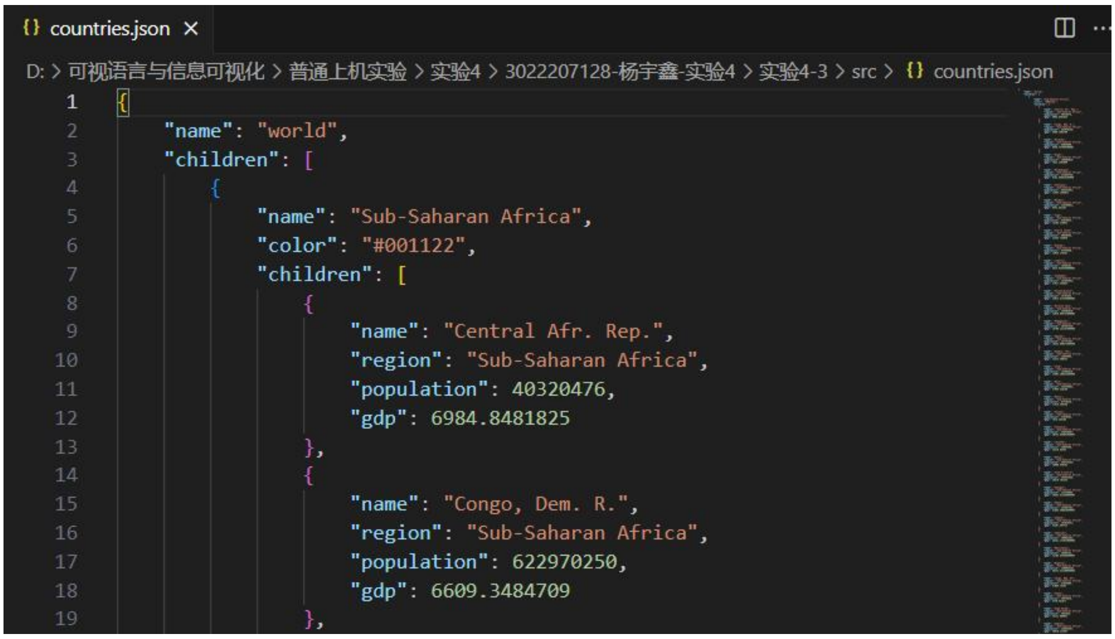
Python 脚本:



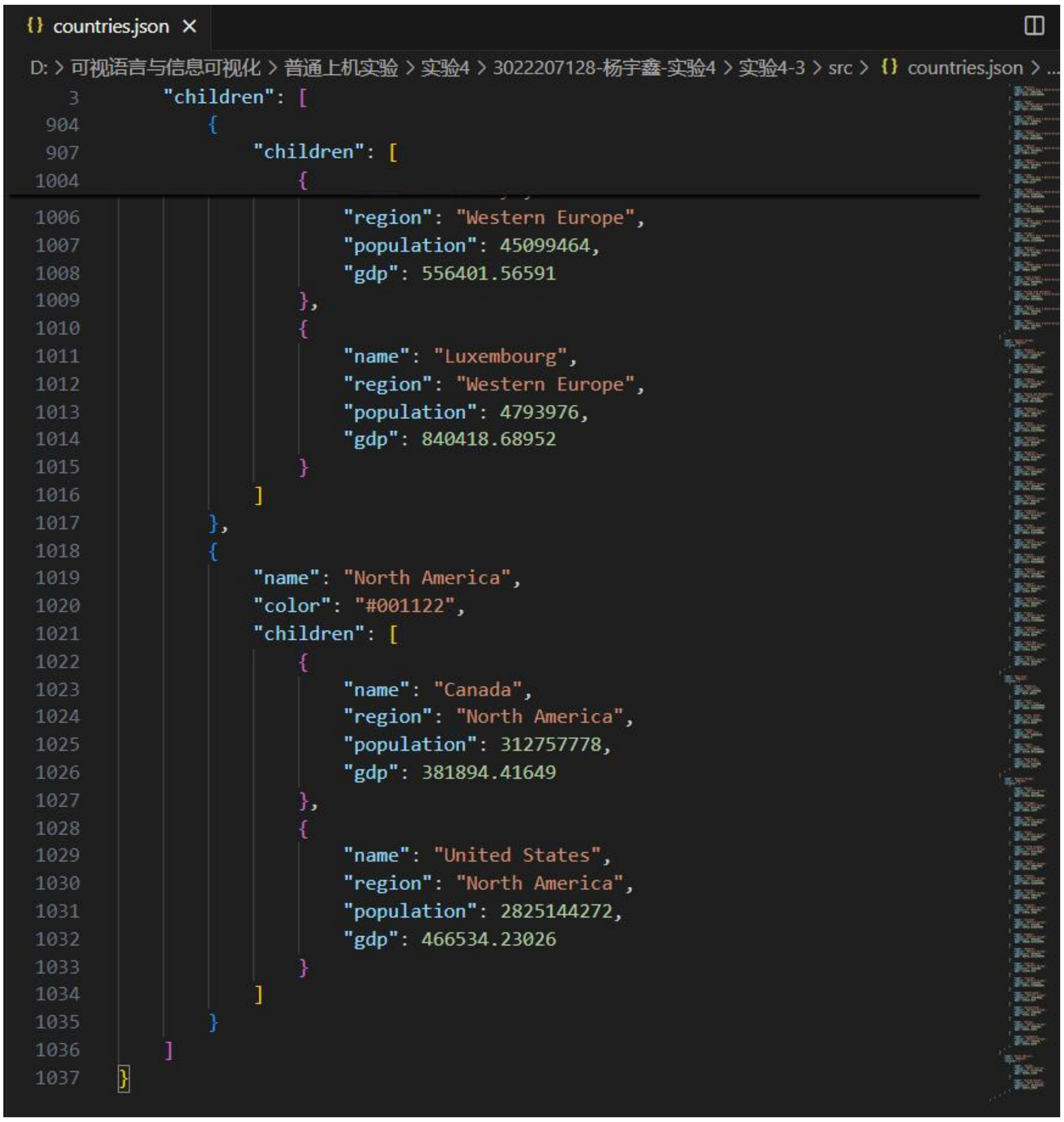
```python
# 用于存储中间结果的字典
data = {}
# 读取CSV文件并处理数据
with open('D:\可视语言与信息可视化\普通上机实验\实验4\\3022207128-杨宇鑫-实验4\实验4-3\src\countries-01.csv', mode='r', encoding='utf-8') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
        # 构建key，以region和countries作为唯一标识
        key = (row["region"], row["countries"])
        if key not in data:
            # 如果key不存在，初始化数据结构
            data[key] = {
                "name": row["countries"],
                "region": row["region"],
                "population": int(row["population"]),
                "gdp": float(row["gdp_ppp_cap"])
            }
        else:
            # 如果key已存在，累加population和gdp
            data[key]["population"] += int(row["population"])
            data[key]["gdp"] += float(row["gdp_ppp_cap"])
# 构建最终的JSON结构
world = {"name": "world", "children": []}
# 将中间结果转换为最终的JSON结构
for item in data.values():
    region_key = item["region"]
    if region_key not in [child["name"] for child in world["children"]]:
        # 如果region不存在，添加新的region节点
        world["children"].append({
            "name": region_key,
            "color": "#" + ''.join([str(hex(int(i))[2:]) for i in range(3) for _ in range(2)]),
            "children": [item]
        })
    else:
        # 如果region已存在，将国家信息添加到对应的region节点下
        for child in world["children"]:
            if child["name"] == region_key:
                child["children"].append(item)
# 将world字典转换为JSON格式并保存到文件
with open('D:\可视语言与信息可视化\普通上机实验\实验4\\3022207128-杨宇鑫-实验4\实验4-3\src\countries.json', 'w', encoding='utf-8') as jsonfile:
    json.dump(world, jsonfile, indent=4, ensure_ascii=False)
```

```python
import csv
import json
# 用于存储中间结果的字典
data = {}
# 读取 CSV 文件并处理数据
with open('D:\可视语言与信息可视化\普通上机实验\实验 4\\3022207128-杨宇鑫-实验 4\
实验 4-3\src\countries-01.csv', mode='r', encoding='utf-8') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
        # 构建 key，以 region 和 countries 作为唯一标识
        key = (row["region"], row["countries"])
        if key not in data:
            # 如果 key 不存在，初始化数据结构
            data[key] = {
                "name": row["countries"],
                "region": row["region"],
                "population": int(row["population"]),
                "gdp": float(row["gdp_ppp_cap"])
            }
        else:
            # 如果 key 已存在，累加 population 和 gdp
            data[key]["population"] += int(row["population"])
            data[key]["gdp"] += float(row["gdp_ppp_cap"])
# 构建最终的 JSON 结构
world = {"name": "world", "children": []}
# 将中间结果转换为最终的 JSON 结构
for item in data.values():
    region_key = item["region"]
    if region_key not in [child["name"] for child in world["children"]]:
        # 如果 region 不存在，添加新的 region 节点
        world["children"].append({
            "name": region_key,
            "color": "#" + ''.join([str(hex(int(i))[2:]) for i in range(3) for
_ in range(2)]),
            "children": [item]
        })
    else:
        # 如果 region 已存在，将国家信息添加到对应的 region 节点下
        for child in world["children"]:
            if child["name"] == region_key:
                child["children"].append(item)
# 将 world 字典转换为 JSON 格式并保存到文件
with open('D:\可视语言与信息可视化\普通上机实验\实验 4\\3022207128-杨宇鑫-实验 4\
实验 4-3\src\countries.json', 'w', encoding='utf-8') as jsonfile:
    json.dump(world, jsonfile, indent=4, ensure_ascii=False)
print("JSON 文件已生成并保存为 'countries.json'")
```

处理结果:


countries.json
D: > 可视语言与信息可视化 > 普通上机实验 > 实验4 > 3022207128-杨宇鑫-实验4 > 实验4-3 > src > countries.json

```
{
    "name": "world",
    "children": [
        {
            "name": "Sub-Saharan Africa",
            "color": "#001122",
            "children": [
                {
                    "name": "Central Afr. Rep.",
                    "region": "Sub-Saharan Africa",
                    "population": 40320476,
                    "gdp": 6984.8481825
                },
                {
                    "name": "Congo, Dem. R.",
                    "region": "Sub-Saharan Africa",
                    "population": 622970250,
                    "gdp": 6609.3484709
                },
```


countries.json
D: > 可视语言与信息可视化 > 普通上机实验 > 实验4 > 3022207128-杨宇鑫-实验4 > 实验4-3 > src > countries.json > ...

```
    "children": [
        {
            "children": [
                {
                    "region": "Western Europe",
                    "population": 45099464,
                    "gdp": 556401.56591
                },
                {
                    "name": "Luxembourg",
                    "region": "Western Europe",
                    "population": 4793976,
                    "gdp": 840418.68952
                }
            ]
        },
        {
            "name": "North America",
            "color": "#001122",
            "children": [
                {
                    "name": "Canada",
                    "region": "North America",
                    "population": 312757778,
                    "gdp": 381894.41649
                },
                {
                    "name": "United States",
                    "region": "North America",
                    "population": 2825144272,
                    "gdp": 466534.23026
                }
            ]
        }
    ]
}
```

2.2 编写代码

2.2.1. HTML 结构

在 `index.html` 和 `index2.html` 文件中，首先定义了基本的 HTML 结构，包括文档类型、头部信息和主体内容。每个文件中都有一个按钮，用于在两个页面之间切换。

```
<button onclick="location.href='index2.html'">人口分布</button>
```

2.2.2. 引入 D3.js 库

在 `<head>` 部分引入 D3.js 及其相关的插件，以便后续进行数据处理和可视化。

```
<script src="https://d3js.org/d3.v6.min.js" charset="utf-8"></script>
<script
src="https://rawcdn.githack.com/Kcnarf/d3-weighted-voronoi/v1.0.1/build/d3-
```

```
weighted-voronoi.js"></script>
<script
src="https://rawcdn.githack.com/Kcnarf/d3-voronoi-map/v2.0.1/build/d3-voron
oi-map.js"></script>
<script
src="https://rawcdn.githack.com/Kcnarf/d3-voronoi-treemap/v1.1.1/build/d3-v
oronoi-treemap.js"></script>
```

2.2.3. CSS 样式

在 `<style>` 标签中定义了一些基本的样式，包括 SVG 的大小、背景颜色、文本样式等。

```
body, html {
  margin: 0;
  overflow: hidden;
}
svg {
  width: 100%;
  height: 100%;
  background-color: rgb(250, 250, 250);
}
```

2.2.4. JavaScript 逻辑

在 `<script>` 标签中，主要实现了以下几个功能：

2.2.4.1 常量定义

定义了一些常量，如 `svgWidth` 和 `svgHeight`，用于设置 SVG 的宽度和高度。

```
var svgWidth = window.innerWidth,
    svgHeight = window.innerHeight;
```

2.2.4.2 数据加载

使用 D3.js 的 `d3.json` 方法加载 `countries.json` 数据文件，并在数据加载完成后初始化数据和布局。

```
d3.json("countries.json").then(function (rootData) {
  initData();
  initLayout(rootData);
});
```

2.2.4.3 数据初始化

在 `initData` 函数中，计算出用于绘制的多边形，并设置字体缩放比例。

```
function initData(rootData) {
  circlingPolygon = computeCirclingPolygon(treemapRadius);
  fontScale.domain([3, 20]).range([8, 20]).clamp(true);
}
```

2.2.4.4 布局初始化

在 initLayout 函数中，创建 SVG 元素和绘图区域，并绘制世界的轮廓。

```
function initLayout(rootData) {
  svg = d3.select("svg")
    .attr("width", svgWidth)
    .attr("height", svgHeight);
  // 绘制世界轮廓
  treemapContainer.append("path")
    .classed("world", true)
    .attr("d", "M" + circlingPolygon.join(",") + "Z");
}
```

2.2.4.5 绘制标题和图例

通过 drawTitle 和 drawLegends 函数绘制标题和图例，帮助用户理解图表内容。

```
function drawTitle() {
  drawingArea.append("text")
    .attr("id", "title")
    .text("2008-2016 世界各国人均 GDP 总值分布");
}
```

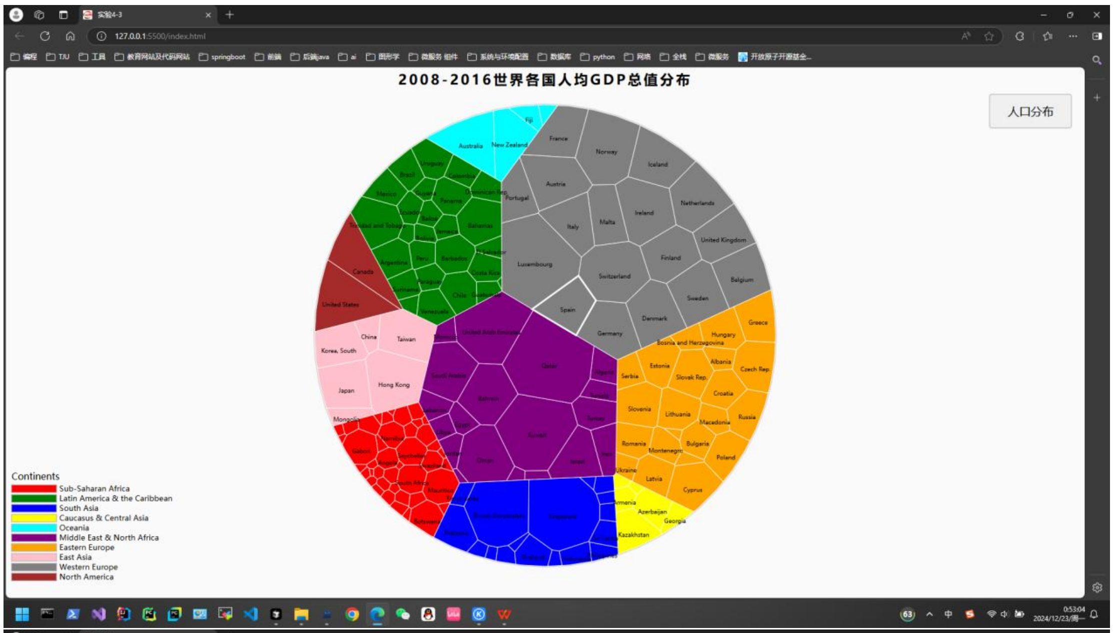
2.2.4.6 绘制树图

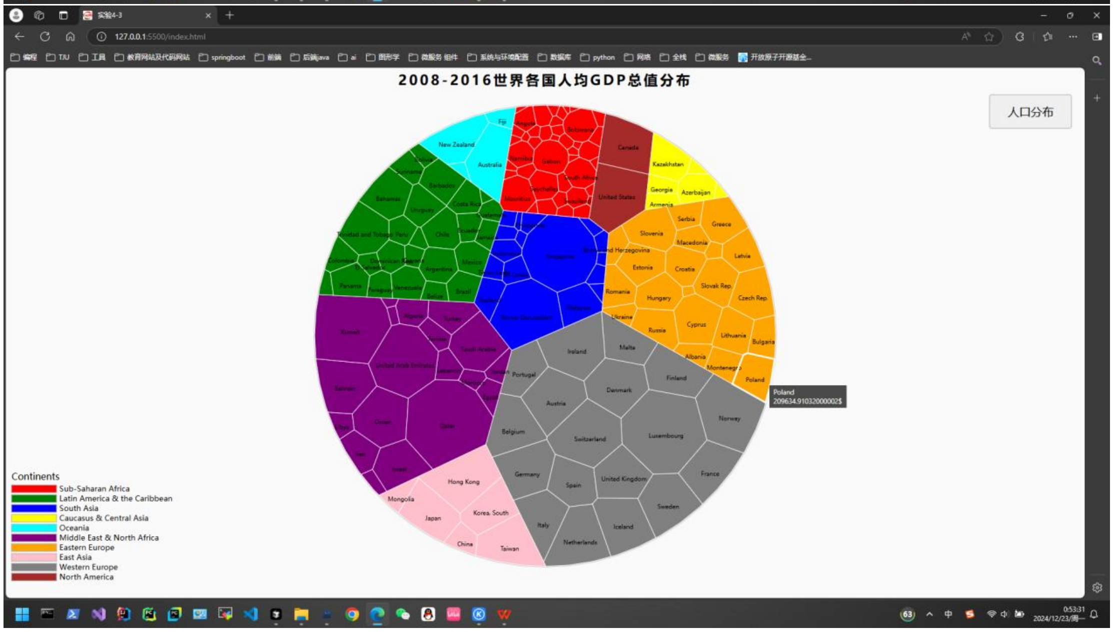
在 drawTreemap 函数中,使用 D3.js 的数据绑定和选择功能绘制树图,展示各国的 GDP 和人口数据。

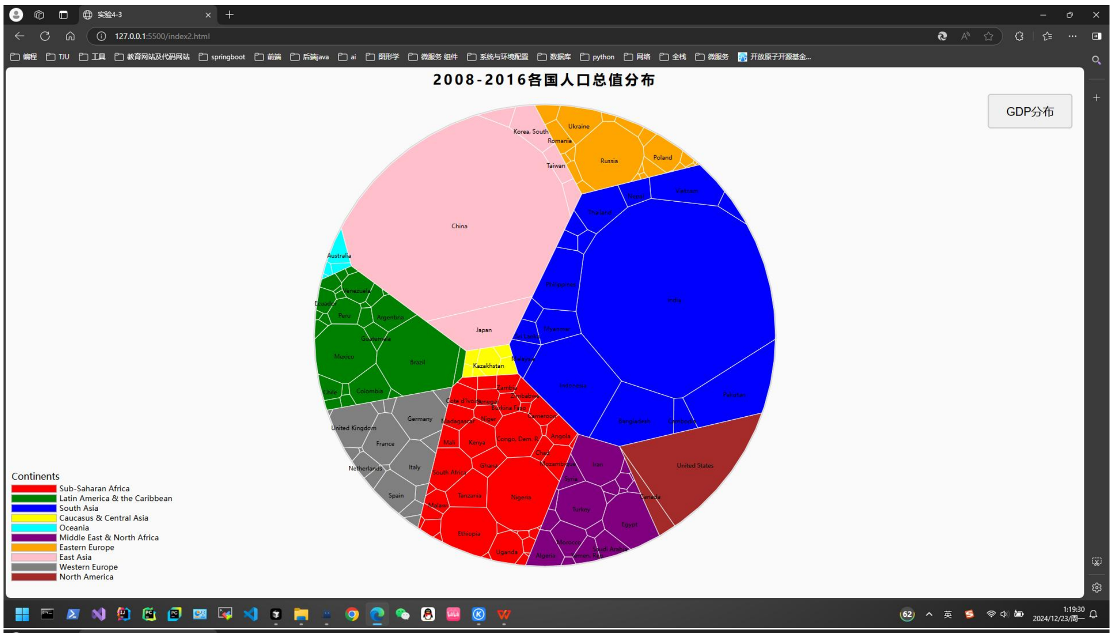
```
function drawTreemap(hierarchy) {
  var leaves = hierarchy.leaves();
  var cells = treemapContainer.append("g")
    .classed('cells', true)
    .selectAll(".cell")
    .data(leaves)
    .enter()
    .append("path")
    .classed("cell", true)
    .style("fill", function (d) {
      return d.parent.data.color;
    });
}
```

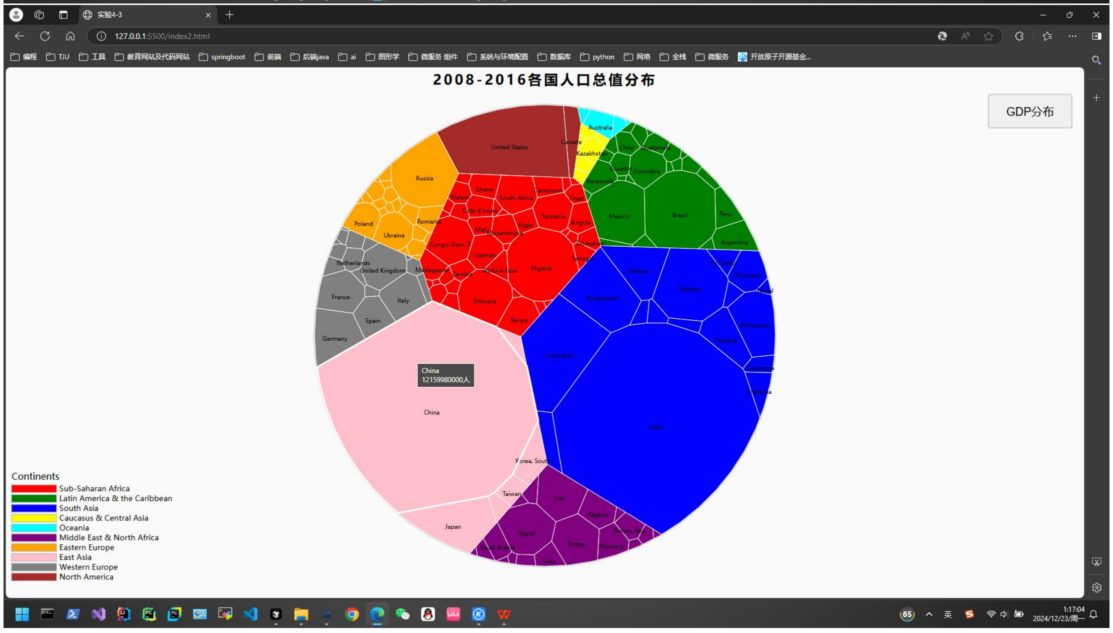
3. 实验结果

通过以上代码实现，用户可以在网页上看到一个动态的可视化地图，展示各国的人均 GDP 和人口分布。用户可以通过点击按钮在不同的页面之间切换，查看不同的数据可视化效果。


2008-2016世界各国人均GDP总值分布
人口分布
Continents
🟥 Sub-Saharan Africa
🟩 Latin America & the Caribbean
🟦 South Asia
🟨 Caucasus & Central Asia
Oceania
🟪 Middle East & North Africa
🟧 Eastern Europe
East Asia
Western Europe
North America




2008-2016各国人口总值分布
GDP分布
Continents
Sub-Saharan Africa
Latin America & the Caribbean
South Asia
Caucasus & Central Asia
Oceania
Middle East & North Africa
Eastern Europe
East Asia
Western Europe
North America
China
India
United States
Japan
Korea, South
Taiwan
Russia
Ukraine
Poland
Romania
Kazakhstan
Australia
Brazil
Mexico
Colombia
Argentina
Peru
Germany
United Kingdom
France
Italy
Spain
Netherlands
Nigeria
Ethiopia
Tanzania
Kenya
South Africa
Ghana
Uganda
Iran
Turkey
Egypt
Pakistan
Bangladesh
Indonesia
Canada


2008-2016各国人口总值分布
GDP分布
Continents
Sub-Saharan Africa
Latin America & the Caribbean
South Asia
Caucasus & Central Asia
Oceania
Middle East & North Africa
Eastern Europe
East Asia
Western Europe
North America
China
1215998000人
India
United States
Russia
Poland
Ukraine
Romania
Australia
Kazakhstan
Canada
Mexico
Brazil
Peru
Colombia
Argentina
Nigeria
Ethiopia
Kenya
Ghana
South Africa
Tanzania
Netherlands
United Kingdom
France
Italy
Spain
Germany
Japan
Korea, South
Taiwan
Iran
Egypt
Turkey
Algeria
Bangladesh
Indonesia