数据科学导论 -HW 4 报告

孙育泉 10234900421
2025.10.14

总览

I 实验要求

对于 ESI 数据的获取方面，可以查看 `hw3` 目录下的文件，不再赘述. 这次我们已经获取到了 `csv` 文件，要求如下:

(1) 将获取的数据导入到一个关系型数据库系统中（系统可以自选）

(2) 优化关系型数据，并整理一个合理的 schema

(3) 通过写 SQL 语句，获取华东师范大学在各个学科中的排名

(4) 通过写 SQL 语句，获取中国（大陆地区）大学在各个学科中的表现

(5) 通过写 SQL 语句，分析全球不同区域在各个学科中的表现

II 具体实现

II.1 建立数据库与导入数据

由于本次实验的任务较轻，且便于 github 的展示，这里选用轻量级的 `SQLite` 作为数据库系统.

首先，引入所需的库文件，检查数据库是否已经存在，如果存在则删除，防止重复插入. 我们先要创建数据库的表结构:

```sql
CREATE TABLE IF NOT EXISTS esi_rankings (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    research_field TEXT NOT NULL,
    rank INTEGER,
    institution TEXT NOT NULL,
    country_region TEXT,
    documents INTEGER,
    cites INTEGER,
    cites_per_paper REAL,
    top_papers INTEGER
);
```

接着，我们将 `csv` 文件中的数据条目插入到数据库中. 在这里使用 `pandas` 读取 `csv` 文件，然后使用 `to_sql` 方法将数据插入到数据库中.

这里要注意的是，所给文件的编码是 `latin-1`，并非 `utf-8`，所以在读取时要指定编码格式. 以及有些数据可能会有空缺，要进行处理.

In [17]:

```python
import sqlite3
import pandas as pd
import os
import glob

csv_folder_path = "./data"  # 包含所有22个CSV文件的文件夹路径
db_path = "./esi_rankings.db"  # 希望生成的数据库文件的存放路径和名称

if os.path.exists(db_path):
    os.remove(db_path)
    print(f"已删除旧的数据库文件: {db_path}")

conn = sqlite3.connect(db_path)
cursor = conn.cursor()

create_table_sql = """
CREATE TABLE IF NOT EXISTS esi_rankings (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        research_field TEXT NOT NULL,
        rank INTEGER,
        institution TEXT NOT NULL,
        country_region TEXT,
        documents INTEGER,
        cites INTEGER,
        cites_per_paper REAL,
        top_papers INTEGER
);
"""
cursor.execute(create_table_sql)
print(f"数据库 '{db_path}' 和数据表 'esi_rankings' 已成功创建. ")

csv_files = glob.glob(os.path.join(csv_folder_path, "*.csv"))

if not csv_files:
    print(f"警告: 在文件夹 '{csv_folder_path}' 中没有找到任何CSV文件. 请检查路径. ")
else:
    for file_path in csv_files:
        file_encoding = "latin-1"

        with open(file_path, "r", encoding=file_encoding) as f:
            first_line = f.readline()

        try:
            field_value = (
                first_line.split("Filter Value(s):")[1].split("Show:")[0].strip()
            )
            print(f"正在处理文件: {os.path.basename(file_path)}, 学科: {field_value}")

            df = pd.read_csv(file_path, header=1, encoding=file_encoding)

            # 重命名列以匹配数据库的Schema
            df.rename(
                columns={
                    "Unnamed: 0": "rank",
```

```
                    "Institutions": "institution",
                    "Countries/Regions": "country_region",
                    "Web of Science Documents": "documents",
                    "Cites": "cites",
                    "Cites/Paper": "cites_per_paper",
                    "Top Papers": "top_papers",
                },
                inplace=True,
            )

            # 关键清洗步骤：移除机构名称为空的无效行（例如CSV末尾的空行）
            initial_rows = len(df)
            df.dropna(subset=["institution"], inplace=True)

            # --- d. 将清洗后的数据写入数据库 ---
            if not df.empty:
                df["research_field"] = field_value
                # 筛选并排序 DataFrame 的列，以确保与数据库表结构一致
                final_df = df[
                    [
                        "research_field",
                        "rank",
                        "institution",
                        "country_region",
                        "documents",
                        "cites",
                        "cites_per_paper",
                        "top_papers",
                    ]
                ]
                # 使用 'append' 模式将数据追加到表中
                final_df.to_sql("esi_rankings", conn, if_exists="append", index=False)
                print(f"✅ 成功导入 {len(final_df)} 条 '{field_value}' 学科的数据。")

        except Exception as e:
            print(f"❌ 处理文件 {os.path.basename(file_path)} 时发生错误: {e}")

# --- 4. 提交更改并关闭数据库连接 ---
conn.commit()
conn.close()
print("\n🎉 所有数据导入完成！数据库 'esi_rankings.db' 已准备就绪。")
```

txt

```
已删除旧的数据库文件: ./esi_rankings.db
数据库 './esi_rankings.db' 和数据表 'esi_rankings' 已成功创建。
正在处理文件: AGRICULTURAL SCIENCES.csv，学科: AGRICULTURAL SCIENCES
✅ 成功导入 1381 条 'AGRICULTURAL SCIENCES' 学科的数据。
正在处理文件: BIOLOGY & BIOCHEMISTRY.csv，学科: BIOLOGY & BIOCHEMISTRY
✅ 成功导入 1649 条 'BIOLOGY & BIOCHEMISTRY' 学科的数据。
正在处理文件: CHEMISTRY.csv，学科: CHEMISTRY
✅ 成功导入 2141 条 'CHEMISTRY' 学科的数据。
正在处理文件: CLINICAL MEDICINE.csv，学科: CLINICAL MEDICINE
✅ 成功导入 6754 条 'CLINICAL MEDICINE' 学科的数据。
正在处理文件: COMPUTER SCIENCE.csv，学科: COMPUTER SCIENCE
✅ 成功导入 863 条 'COMPUTER SCIENCE' 学科的数据。
正在处理文件: ECONOMICS & BUSINESS.csv，学科: ECONOMICS & BUSINESS
✅ 成功导入 543 条 'ECONOMICS & BUSINESS' 学科的数据。
```

```
正在处理文件: ENGINEERING.csv, 学科: ENGINEERING
✅ 成功导入 2787 条 'ENGINEERING' 学科的数据.
正在处理文件: ENVIRONMENT ECOLOGY.csv, 学科: ENVIRONMENT/ECOLOGY
✅ 成功导入 2066 条 'ENVIRONMENT/ECOLOGY' 学科的数据.
正在处理文件: GEOSCIENCES.csv, 学科: GEOSCIENCES
✅ 成功导入 1175 条 'GEOSCIENCES' 学科的数据.
正在处理文件: IMMUNOLOGY.csv, 学科: IMMUNOLOGY
✅ 成功导入 1177 条 'IMMUNOLOGY' 学科的数据.
正在处理文件: MATERIALS SCIENCE.csv, 学科: MATERIALS SCIENCE
✅ 成功导入 1580 条 'MATERIALS SCIENCE' 学科的数据.
正在处理文件: MATHEMATICS.csv, 学科: MATHEMATICS
✅ 成功导入 395 条 'MATHEMATICS' 学科的数据.
正在处理文件: MICROBIOLOGY.csv, 学科: MICROBIOLOGY
✅ 成功导入 803 条 'MICROBIOLOGY' 学科的数据.
正在处理文件: MOLECULAR BIOLOGY & GENETICS.csv, 学科: MOLECULAR BIOLOGY & GENETICS
✅ 成功导入 1169 条 'MOLECULAR BIOLOGY & GENETICS' 学科的数据.
正在处理文件: MULTIDISCIPLINARY.csv, 学科: MULTIDISCIPLINARY
✅ 成功导入 216 条 'MULTIDISCIPLINARY' 学科的数据.
正在处理文件: NEUROSCIENCE & BEHAVIOR.csv, 学科: NEUROSCIENCE & BEHAVIOR
✅ 成功导入 1298 条 'NEUROSCIENCE & BEHAVIOR' 学科的数据.
正在处理文件: PHARMACOLOGY & TOXICOLOGY.csv, 学科: PHARMACOLOGY & TOXICOLOGY
✅ 成功导入 1389 条 'PHARMACOLOGY & TOXICOLOGY' 学科的数据.
正在处理文件: PHYSICS.csv, 学科: PHYSICS
✅ 成功导入 995 条 'PHYSICS' 学科的数据.
正在处理文件: PLANT & ANIMAL SCIENCE.csv, 学科: PLANT & ANIMAL SCIENCE
✅ 成功导入 1950 条 'PLANT & ANIMAL SCIENCE' 学科的数据.
正在处理文件: PSYCHIATRY PSYCHOLOGY.csv, 学科: PSYCHIATRY/PSYCHOLOGY
✅ 成功导入 1147 条 'PSYCHIATRY/PSYCHOLOGY' 学科的数据.
正在处理文件: SOCIAL SCIENCES, GENERAL.csv, 学科: SOCIAL SCIENCES, GENERAL
✅ 成功导入 2407 条 'SOCIAL SCIENCES, GENERAL' 学科的数据.
正在处理文件: SPACE SCIENCE.csv, 学科: SPACE SCIENCE
✅ 成功导入 236 条 'SPACE SCIENCE' 学科的数据.

🎉 所有数据导入完成！数据库 'esi_rankings.db' 已准备就绪.

```

至此，创建数据库工作完成.

II.2 选择合适的 Schema

根据这个项目要求和数据特点，一个“好”的结构应该是：

(1) **逻辑清晰**：能够直观地反映数据.

(2) **查询高效**：能快速地执行分析查询.

(3) **避免冗余**：在合理的范围内减少重复数据.

(4) **保证数据完整性**：确保存入的数据是有效和一致的.

先来看一下现在的 schema：

列名 (Column Name)	数据类型 (Data Type)	约束/说明 (Constraints/Notes)
`id`	`INTEGER`	主键 *(PRIMARY KEY)*, `AUTOINCREMENT`. 每一行的唯一标识，数据库自动生成.

列名 (Column Name)	数据类型 (Data Type)	约束/说明 (Constraints/Notes)
research_field	TEXT	不能为空 *(NOT NULL)*. 关键的分类字段, 如 “CHEMISTRY”, “ENGINEERING”.
rank	INTEGER	机构在该学科下的排名.
institution	TEXT	不能为空 *(NOT NULL)*. 机构名称.
country_region	TEXT	国家或地区.
documents	INTEGER	Web of Science 文献数.
cites	INTEGER	总引用数.
cites_per_paper	REAL	篇均引用数. 使用 REAL (浮点数) 来精确存储小数.
top_papers	INTEGER	顶尖论文数.

事实上，这样的安排已经很好了. 虽然，我们后续要查询“机构”或者“国家”，但是这些字段并不适合单独拆分成表格，因为它们的种类太多，且没有明显的层级关系. 拆分成表格反而会增加查询的复杂度，在这张表中，我们通过简单的 WHERE 和 GROUP BY 就可以完成. 如果分表，就需要使用 JOIN 操作，这会让查询变得更复杂.

II.3 获取 ECNU 在各个学科中的排名

为了方便后续的代码编写，我们先定义如下几个东西:

- execute_query(): 执行 SQL 查询的函数
- get_iso_alpha_3(): 获取国家的 ISO Alpha-3 代码的函数，用于最后一步实现全球可视化
- DATABASE_TITLE: 数据库的路径

In [18]:

```python
import sqlite3
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import pycountry


def execute_query(db_path, query):
    """
        功能: 连接指定的SQLite数据库, 执行一条SQL查询语句.
        参数:
                db_path (str): 数据库文件的路径.
                query (str): 需要执行的SQL查询字符串.
        返回:
                pandas.DataFrame: 包含查询结果的DataFrame. 如果出错则返回None.
        """
    try:
        with sqlite3.connect(db_path) as conn:
            df = pd.read_sql_query(query, conn)
            return df
    except Exception as e:
        print(f"执行查询时发生错误: {e}")
        return None
```

```python

def get_iso_alpha_3(country_name):
    """
        功能：将国家/地区名称转换为标准的ISO 3166-1 alpha-3三字符代码。
        说明：内置了一个手动映射表来处理数据库中的非标准名称。
        参数：
                country_name (str): 数据库中的国家/地区名称。
        返回：
                str: 对应的三字符ISO代码，如果找不到则返回None。
        """
    # 手动映射表，处理 'pycountry' 无法直接识别的特殊名称
    manual_map = {
        "CHINA MAINLAND": "CHN",
        "USA": "USA",
        "ENGLAND": "GBR",  # 英格兰、苏格兰等均属于联合王国 (GBR)
        "SCOTLAND": "GBR",
        "WALES": "GBR",
        "NORTH IRELAND": "GBR",
        "GERMANY (FED REP GER)": "DEU",
        "SOUTH KOREA": "KOR",
        "RUSSIA": "RUS",
        "IRAN": "IRN",
    }

    if country_name in manual_map:
        return manual_map[country_name]

    try:
        # 尝试使用pycountry的模糊搜索功能来查找匹配项
        return pycountry.countries.search_fuzzy(country_name)[0].alpha_3
    except (LookupError, AttributeError):
        # 如果找不到任何匹配，则返回None
        return None


# --- 全局设置 ---
# 定义数据库文件路径，后续所有单元格都将使用此变量
DATABASE_FILE = "esi_rankings.db"
print("✅ 环境准备就绪，核心函数已定义。")
```

txt

```
✅ 环境准备就绪，核心函数已定义。

```

这里的处理是，为了方便后续的对比，我们定义了一个所查机构的列表，用于遍历．只需要在查询到 ECNU 的时候进行输出即可．这里的查询语句就是看机构的名字是否与 `institution` 字段匹配．注意，我们使用了 `LIKE` 语句，而不是 `=`，为了提高我们查询的容错．

In [19]:

python

```python
universities_to_compare = [
    "EAST CHINA NORMAL UNIV",
    "PEKING UNIV",
    "TSINGHUA UNIV",
    "FUDAN UNIV",
    "TONGJI UNIV",
```

```
    "SHANGHAI JIAO TONG UNIV",
]

all_rankings_data = []
print("正在从数据库中查询指定大学的排名数据...")

for university in universities_to_compare:
    query = f"SELECT research_field, rank FROM esi_rankings WHERE institution LIKE '%{university}%'"
    df = execute_query(DATABASE_FILE, query)
    if df is not None and not df.empty:
        df["institution"] = university  # 添加大学名称列用于分组
        if university == universities_to_compare[0]:
            print(df)
        all_rankings_data.append(df)
```

txt

```
正在从数据库中查询指定大学的排名数据...
                research_field  rank             institution
0          AGRICULTURAL SCIENCES   845  EAST CHINA NORMAL UNIV
1         BIOLOGY & BIOCHEMISTRY   721  EAST CHINA NORMAL UNIV
2                      CHEMISTRY    90  EAST CHINA NORMAL UNIV
3             CLINICAL MEDICINE  2852  EAST CHINA NORMAL UNIV
4               COMPUTER SCIENCE   207  EAST CHINA NORMAL UNIV
5                    ENGINEERING   317  EAST CHINA NORMAL UNIV
6           ENVIRONMENT/ECOLOGY   130  EAST CHINA NORMAL UNIV
7                    GEOSCIENCES   275  EAST CHINA NORMAL UNIV
8             MATERIALS SCIENCE   196  EAST CHINA NORMAL UNIV
9                    MATHEMATICS   115  EAST CHINA NORMAL UNIV
10  MOLECULAR BIOLOGY & GENETICS   867  EAST CHINA NORMAL UNIV
11      NEUROSCIENCE & BEHAVIOR   853  EAST CHINA NORMAL UNIV
12    PHARMACOLOGY & TOXICOLOGY  1064  EAST CHINA NORMAL UNIV
13                       PHYSICS   522  EAST CHINA NORMAL UNIV
14        PLANT & ANIMAL SCIENCE   395  EAST CHINA NORMAL UNIV
15         PSYCHIATRY/PSYCHOLOGY   467  EAST CHINA NORMAL UNIV
16     SOCIAL SCIENCES, GENERAL   314  EAST CHINA NORMAL UNIV

```

为了方便对比，这里罗列了一些高校在所有学科中的排名.（注意：越低越好 hhh）

In [20]:

python

```
if all_rankings_data:
    # 将所有查询结果合并成一个DataFrame
    comparison_df = pd.concat(all_rankings_data, ignore_index=True)
    print("数据获取完毕，正在生成可视化图表...")

    # 使用Seaborn绘制分组条形图
    plt.figure(figsize=(20, 12))
    sns.set_theme(style="whitegrid", font="Maple Mono Normal NF CN")  # 设置支持中文的字体，如黑体
    plt.rcParams["axes.unicode_minus"] = False  # 解决负号显示问题

    sns.barplot(
        data=comparison_df,
        x="research_field",
        y="rank",
        hue="institution",
```

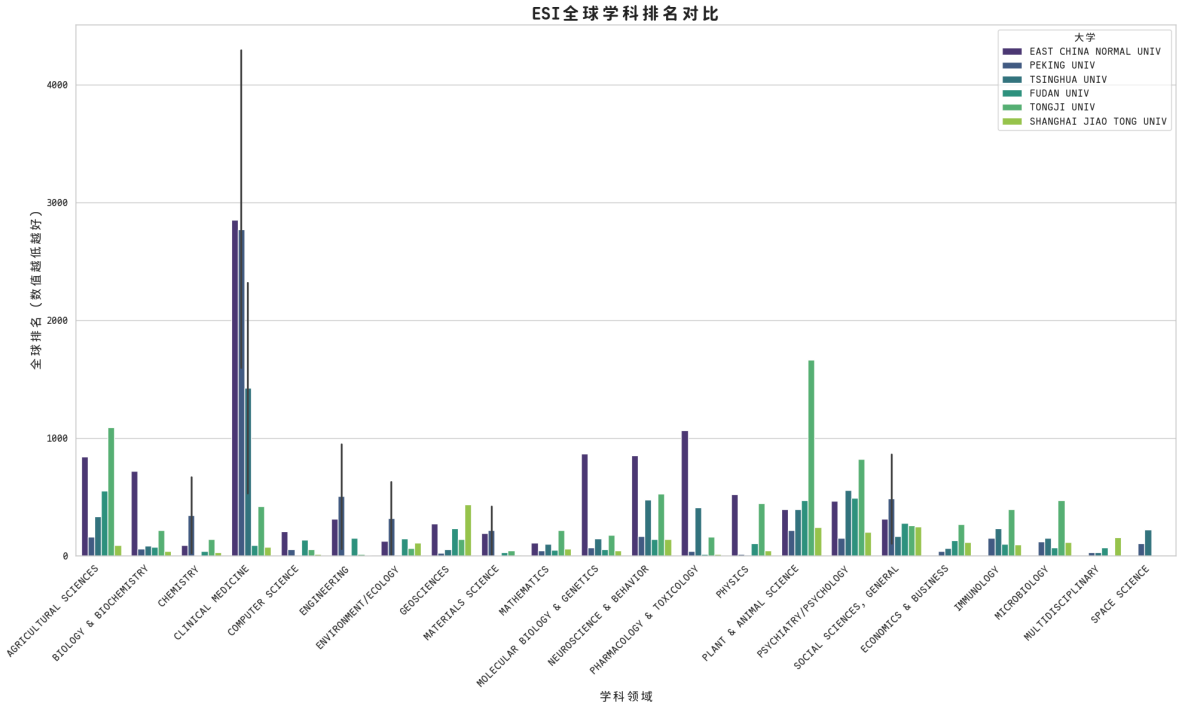
```
        palette="viridis",
    )

    # 美化图表
    plt.title("ESI全球学科排名对比", fontsize=20, weight="bold")
    plt.xlabel("学科领域", fontsize=14)
    plt.ylabel("全球排名（数值越低越好）", fontsize=14)
    plt.xticks(rotation=45, ha="right", fontsize=12)
    plt.legend(title="大学", fontsize=12)
    plt.tight_layout()
    plt.show()
else:
    print("未能查询到任何指定大学的数据. ")
```

txt

```
数据获取完毕，正在生成可视化图表...

```



II.4 获取中国（大陆地区）大学在各个学科中的表现

这里，我们使用了 `country_region` 字段来筛选出中国（大陆地区）的大学. 然后，我们考虑两种指标:

(1) 在每个学科中，有多少所中国大陆的大学进入了全球排名前 100 名

(2) 在每个学科中，排名第一的中国大陆大学

既注重整体，也注重代表.

In [21]:

python

```
print("--- 分析中国（大陆）大学在各学科的表现 ---")

# --- 分析角度1: 集体实力 ---
```

```txt
# 统计在每个学科中，有多少所中国大陆的大学进入了全球排名前100名.
# 这反映了中国在哪些学科领域具有广泛的高水平研究基础.
print("\n>>> 分析1: 各学科领域进入全球前100的中国（大陆）大学数量")
query_strength = """
        SELECT
            research_field AS '学科领域',
            COUNT(institution) AS '全球前100大学数量'
        FROM
            esi_rankings
        WHERE
            country_region = 'CHINA MAINLAND' AND rank <= 100
        GROUP BY
            research_field
        ORDER BY
            '全球前100大学数量' DESC;
"""
df_strength = execute_query(DATABASE_FILE, query_strength)
if df_strength is not None:
    # 使用 display() 函数在Jupyter中可以获得更好的表格格式化效果
    display(df_strength)
else:
    print("未能查询到相关数据. ")


# --- 分析角度2: 顶尖实力 ---
# 找出在每个学科中，排名第一的中国大陆大学.
# 这反映了中国在各个学科领域的"领头羊".
print("\n>>> 分析2: 各学科领域排名第一的中国（大陆）大学")
query_leaders = """
        SELECT
            research_field AS '学科领域',
            institution AS '排名第一的大学',
            rank AS '全球排名'
        FROM
            (
                SELECT
                    research_field,
                    institution,
                    rank,
                    ROW_NUMBER() OVER(PARTITION BY research_field ORDER BY rank ASC)
              as rn
                FROM
                    esi_rankings
                WHERE
                    country_region = 'CHINA MAINLAND'
            )
        WHERE
            rn = 1
        ORDER BY
            research_field ASC;
"""
df_leaders = execute_query(DATABASE_FILE, query_leaders)
if df_leaders is not None:
    display(df_leaders)
else:
    print("未能查询到相关数据. ")
```

```
--- 分析中国（大陆）大学在各学科的表现 ---

>>> 分析1: 各学科领域进入全球前100的中国（大陆）大学数量

```

txt

	学科领域	全球前100大学数量
0	SPACE SCIENCE	2
1	SOCIAL SCIENCES, GENERAL	1
2	PLANT & ANIMAL SCIENCE	13
3	PHYSICS	10
4	PHARMACOLOGY & TOXICOLOGY	29
5	NEUROSCIENCE & BEHAVIOR	2
6	MULTIDISCIPLINARY	5
7	MOLECULAR BIOLOGY & GENETICS	9
8	MICROBIOLOGY	11
9	MATHEMATICS	24
10	MATERIALS SCIENCE	48
11	IMMUNOLOGY	4
12	GEOSCIENCES	22
13	ENVIRONMENT/ECOLOGY	14
14	ENGINEERING	46
15	ECONOMICS & BUSINESS	8
16	COMPUTER SCIENCE	44
17	CLINICAL MEDICINE	3
18	CHEMISTRY	46
19	BIOLOGY & BIOCHEMISTRY	8
20	AGRICULTURAL SCIENCES	22

txt

```

>>> 分析2: 各学科领域排名第一的中国（大陆）大学

```

txt

	学科领域	排名第一的大学	全球排名
0	AGRICULTURAL SCIENCES	CHINESE ACADEMY OF SCIENCES	1
1	BIOLOGY & BIOCHEMISTRY	CHINESE ACADEMY OF SCIENCES	3
2	CHEMISTRY	CHINESE ACADEMY OF SCIENCES	1
3	CLINICAL MEDICINE	SHANGHAI JIAO TONG UNIVERSITY	75
4	COMPUTER SCIENCE	CHINESE ACADEMY OF SCIENCES	1
5	ECONOMICS & BUSINESS	PEKING UNIVERSITY	44
6	ENGINEERING	CHINESE ACADEMY OF SCIENCES	1
7	ENVIRONMENT/ECOLOGY	CHINESE ACADEMY OF SCIENCES	1
8	GEOSCIENCES	CHINESE ACADEMY OF SCIENCES	1
9	IMMUNOLOGY	CHINESE ACADEMY OF SCIENCES	36
10	MATERIALS SCIENCE	CHINESE ACADEMY OF SCIENCES	1
11	MATHEMATICS	CHINESE ACADEMY OF SCIENCES	4
12	MICROBIOLOGY	CHINESE ACADEMY OF SCIENCES	3
13	MOLECULAR BIOLOGY & GENETICS	CHINESE ACADEMY OF SCIENCES	13
14	MULTIDISCIPLINARY	CHINESE ACADEMY OF SCIENCES	3
15	NEUROSCIENCE & BEHAVIOR	CHINESE ACADEMY OF SCIENCES	71
16	PHARMACOLOGY & TOXICOLOGY	CHINESE ACADEMY OF SCIENCES	4
17	PHYSICS	CHINESE ACADEMY OF SCIENCES	1
18	PLANT & ANIMAL SCIENCE	CHINESE ACADEMY OF SCIENCES	1
19	PSYCHIATRY/PSYCHOLOGY	PEKING UNIVERSITY	154
20	SOCIAL SCIENCES, GENERAL	CHINESE ACADEMY OF SCIENCES	47
21	SPACE SCIENCE	CHINESE ACADEMY OF SCIENCES	15

II.5 分析全球不同区域在各个学科中的表现

这里，为了以地区为单位，我们在查询的时候，使用了 `country_region` 字段进行分组. 我们统计了每个国家在各个学科中的总引用数并进行了可视化.

python

```python
print("正在获取全球各学科数据用于生成世界地图...")
subject_data_query = """
        SELECT
            country_region,
            research_field,
            SUM(cites) AS total_cites
        FROM
            esi_rankings
        GROUP BY
            country_region, research_field;
"""
df_map = execute_query(DATABASE_FILE, subject_data_query)

if df_map is not None:
    print("数据预处理中：清洗、转换ISO代码...")
    df_map.dropna(subset=["country_region", "research_field"], inplace=True)
    df_map["total_cites"] = pd.to_numeric(
        df_map["total_cites"], errors="coerce"
    ).fillna(0)
    df_map["iso_alpha"] = df_map["country_region"].apply(
        get_iso_alpha_3
    )  # 调用单元格1的函数

    subjects = sorted(df_map["research_field"].unique())

    print("正在生成交互式地图...")
    fig = go.Figure()

    for subject in subjects:
        subject_df = df_map[df_map["research_field"] == subject]
        log_cites = np.log10(
            subject_df["total_cites"] + 1
        )  # 对引用数取对数，以优化颜色显示

        fig.add_trace(
            go.Choropleth(
                locations=subject_df["iso_alpha"],
                z=log_cites,
                text=subject_df["country_region"],
                customdata=subject_df["total_cites"],
                hovertemplate="<b>%{text}</b><br>总引用数: %{customdata:,.0f}<extra></
                extra>",
                colorscale="Plasma",
                visible=(subject == subjects[0]),  # 默认只显示第一个学科
            )
        )

    buttons = []
    for i, subject in enumerate(subjects):
        visibility_mask = [False] * len(subjects)
        visibility_mask[i] = True
        buttons.append(
            dict(label=subject, method="update", args=[{"visible": visibility_mask}])
        )
```

```
54
55      fig.update_layout(
56          title_text=f"<b>全球各学科学术影响力地图</b><br><i>请从下拉菜单中选择一个学科</i>",
57          title_x=0.5,
58          title_font_size=22,
59          margin=dict(t=70, l=0, r=0, b=0),
60          updatemenus=[
61              dict(
62                  active=0,
63                  buttons=buttons,
64                  direction="down",
65                  pad={"r": 10, "t": 10},
66                  showactive=True,
67                  x=0.1,  # 将菜单的x坐标设置在左侧10%的位置
68                  xanchor="left",  # x坐标的锚点是菜单的左边缘
69                  y=1.1,  # 将菜单的y坐标放在新的顶部空间内
70                  yanchor="top",  # y坐标的锚点是菜单的上边缘
71              )
72          ],
73          geo=dict(
74              showframe=False, showcoastlines=False, projection_type="natural earth"
75          ),
76      )
77
78      fig.show()
79  else:
80      print("未能获取用于生成地图的数据. ")
```

这里只截取了 CS 学科的结果截图:

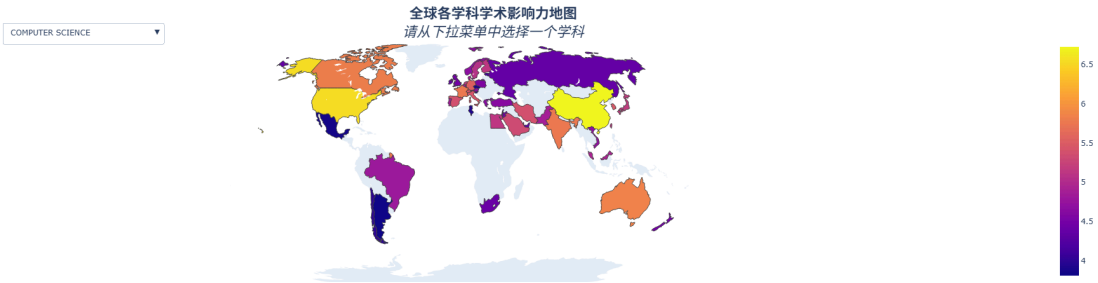


Figure 1: 全球机构 - computer science

> **Remark**
>
> - 完整代码见 `hw4.ipynb`.
> - 最后一部分为交互图表，在 `pdf` 文件中无法展示，可以打开 `jupyter notebook` 查看.