基于 Covid-19 传播数据的 Spark 数据处理分析

1. 启动 hadoop 并上传数据

```
hadoop@master:~$ cd /usr/local/hadoop-3.1.3
bash: cd: /usr/local/hadoop-3.1.3: 没有那个文件或目录
hadoop@master:~$ ./sbin/start-dfs.sh
bash: ./sbin/start-dfs.sh: 没有那个文件或目录
hadoop@master:~$ cd /usr/local/hadoop
hadoop@master:/usr/local/hadoop$ ./sbin/start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [master]
hadoop@master:/usr/local/hadoop$ bin/hadoop fs -mkdir -p /dbcovid/data
mkdir: Cannot create directory /dbcovid/data. Name node is in safe mode.
hadoop@master:/usr/local/hadoop$ sudo bin/hadoop fs -mkdir -p /dbcovid/data
[sudo] hadoop 的密码:
mkdir: Permission denied: user=root, access=WRITE, inode="/":hadoop:supergroup:d
rwxr-xr-x
hadoop@master:/usr/local/hadoop$ ^C
hadoop@master:/usr/local/hadoop$ bin/hadoop fs -mkdir -p /dbcovid/data
hadoop@master:/usr/local/hadoop$ bin/hadoop fs -put ~/下载/covid.csv /dbcovid/da
ta
hadoop@master:/usr/local/hadoop$ /usr/local/hadoop-3.1.3/bin/hadoop fs -ls /dbco
vid/data/covid.csv
bash: /usr/local/hadoop-3.1.3/bin/hadoop: 没有那个文件或目录
hadoop@master:/usr/local/hadoop$ /usr/local/hadoop/bin/hadoop fs -ls /dbcovid/da
ta/covid.csv
```

2. 数据预处理

```
scala> val datapath = "hdfs://localhost:9000/dbcovid/data/covid.csv"
datapath: String = hdfs://localhost:9000/dbcovid/data/covid.csv

scala> val df = spark.read.option("header", "true").option("inferSchema","true")
.csv(datapath)
2025-05-10 12:14:45,249 INFO internal.SharedState: Setting hive.metastore.wareho
use.dir ('null') to the value of spark.sql.warehouse.dir.
2025-05-10 12:14:45,272 INFO internal.SharedState: Warehouse path is 'file:/usr/
local/hadoop/spark-warehouse'.
2025-05-10 12:14:45,301 INFO handler.ContextHandler: Started o.s.j.s.ServletCont
extHandler@734990c1{/SQL,null,AVAILABLE,@Spark}
2025-05-10 12:14:45,302 INFO handler.ContextHandler: Started o.s.j.s.ServletCont
extHandler@4f7db06c{/SQL/json,null,AVAILABLE,@Spark}
2025-05-10 12:14:45,304 INFO handler.ContextHandler: Started o.s.j.s.ServletCont
extHandler@651a3e01{/SQL/execution,null,AVAILABLE,@Spark}
2025-05-10 12:14:45,305 INFO handler.ContextHandler: Started o.s.j.s.ServletCont
extHandler@6870f52a{/SQL/execution/json,null,AVAILABLE,@Spark}
2025-05-10 12:14:45,335 INFO handler.ContextHandler: Started o.s.j.s.ServletCont
extHandler@224bff6{/static/sql,null,AVAILABLE,@Spark}
```

```
scala> df.printSchema()
root
 |-- iso_code: string (nullable = true)
 |-- continent: string (nullable = true)
 |-- location: string (nullable = true)
 |-- date: date (nullable = true)
 |-- total_cases: double (nullable = true)
 |-- new_cases: double (nullable = true)
 |-- new_cases_smoothed: double (nullable = true)
 |-- total_deaths: double (nullable = true)
 |-- new_deaths: double (nullable = true)
 |-- new_deaths_smoothed: double (nullable = true)
 |-- total_cases_per_million: double (nullable = true)
 |-- new_cases_per_million: double (nullable = true)
 |-- new_cases_smoothed_per_million: double (nullable = true)
 |-- total_deaths_per_million: double (nullable = true)
 |-- new_deaths_per_million: double (nullable = true)
 |-- new_deaths_smoothed_per_million: double (nullable = true)
 |-- reproduction_rate: double (nullable = true)
 |-- icu_patients: double (nullable = true)
 |-- icu_patients_per_million: double (nullable = true)
 |-- hosp_patients: double (nullable = true)
 |-- hosp_patients_per_million: double (nullable = true)
```

3. 使用 Spark 进行数据分析

基本统计分析

```
2025-05-10 12:16:54,461 INFO codegen.CodeGenerator:
ed in 40.822258 ms
+-------------+
|    continent|
+-------------+
|       Europe|
|       Africa|
|         null|
|North America|
|South America|
|      Oceania|
|         Asia|
+-------------+
```

统计国家数量

```
ed in 23.626134 ms
+------------------------+
|count(DISTINCT location)|
+------------------------+
|                     238|
+------------------------+
```

统计日期总数

```
+-----------+
|count(date)|
+-----------+
|        774|
+-----------+
```

独立指标分析

```
scala> val locs = List("China", "United States", "European Union
", "Russia", "Japan", "United Kingdom", "Singapore")
2025-05-10 12:18:37,816 INFO storage.BlockManagerInfo: Removed b
roadcast_13_piece0 on 192.168.128.148:39243 in memory (size: 5.9
 KiB, free: 366.2 MiB)
locs: List[String] = List(China, United States, European Union,
Russia, Japan, United Kingdom, Singapore)
```

（1）（重点国家）新增病例/死亡数量

```
scala> for (loc <- locs) {
     |   spark.sql(s"select new_cases from data where location='
$loc'").write.json(s"/dbcovid/result/new_cases/$loc/")
     |   spark.sql(s"select new_deaths from data where location=
'$loc'").write.json(s"/dbcovid/result/new_deaths/$loc/")
     | }
2025-05-10 12:20:19,719 INFO datasources.FileSourceStrategy: Pus
hed Filters: IsNotNull(location),EqualTo(location,China)
2025-05-10 12:20:19,719 INFO datasources.FileSourceStrategy: Pos
t-Scan Filters: isnotnull(location#19),(location#19 = China)
2025-05-10 12:20:19,813 INFO output.PathOutputCommitterFactory:
```

（2）（重点国家）累计病例/死亡数量

```
scala> for (loc <- locs) {
     |   spark.sql(s"select total_cases from data where location
='$loc'").write.json(s"/dbcovid/result/total_cases/$loc/")
     |   spark.sql(s"select total_deaths from data where locatio
n='$loc'").write.json(s"/dbcovid/result/total_deaths/$loc/")
     | }
```

（3）（重点国家）疫苗接种剂次

```
scala> for (loc <- locs) {
     |   spark.sql(s"select total_vaccinations from data where l
ocation='$loc'").write.json(s"/dbcovid/result/total_vacc/$loc/")

     | }
```

（4）（全部国家）每百万人累计病例/死亡数量

```
scala> spark.sql("select location,max(1000000*total_deaths/popul
ation) from data where continent is not null group by location")
.write.json("/dbcovid/result/million_deaths/")
```

相关性分析

（5）（重点国家）每日新增病例数与疫苗接种剂次的关系

```
scala> for (loc <- locs) {
     |   spark.sql(s"select new_cases,total_vaccinations from da
ta where location='$loc'").write.json(s"/dbcovid/result/cases_wi
th_vaccs/$loc/")
     | }
```

（6）（所有国家）每百万人病例/死亡数量与其他指标的关系

```
scala> spark.sql("select location,max(1000000*total_deaths/popul
ation),max(gdp_per_capita) from data where continent is not null
 group by location").write.json("/dbcovid/result/million_deaths_
with_gdp/")
```

```
scala> spark.sql("select location,max(1000000*total_deaths/popul
ation),max(median_age) from data where continent is not null gro
up by location").write.json("/dbcovid/result/million_deaths_with
_age/")
```

（7）每百万人病例数量与人均 GDP 关系

```
scala> spark.sql("select location,max(1000000*total_cases/population),max(gdp_per_capita) from data where continent is not
 null group by location").write.json("/dbcovid/result/million_cases_with_gdp/")
2025-05-12 03:02:05,746 INFO storage.BlockManagerInfo: Removed broadcast_2_piece0 on 192.168.128.150:45327 in memory (size
: 53.3 KiB, free: 366.3 MiB)
2025-05-12 03:02:05,798 INFO storage.BlockManagerInfo: Removed broadcast_3_piece0 on 192.168.128.150:45327 in memory (size
: 13.0 KiB, free: 366.3 MiB)
2025-05-12 03:02:06,108 INFO datasources.FileSourceStrategy: Pushed Filters: IsNotNull(continent)
2025-05-12 03:02:06,109 INFO datasources.FileSourceStrategy: Post-Scan Filters: isnotnull(continent#18)
2025-05-12 03:02:06,627 INFO codegen.CodeGenerator: Code generated in 224.648065 ms
```

每百万人死亡数量与人均 GDP 关系

```
scala> spark.sql("select location,max(1000000*total_deaths/population),max(gdp_per_capita) from data where continent is no
t null group by location").write.json("/dbcovid/result/million_deaths_with_gdp/")
```

（8）（所有国家）每百万人病例数量与年龄中位数关系

```
scala> spark.sql("select location,max(1000000*total_cases/population),max(median_age) from data where continent is not nul
l group by location").write.json("/dbcovid/result/million_cases_with_age/")
2025-05-12 03:04:58,801 INFO datasources.FileSourceStrategy: Pushed Filters: IsNotNull(continent)
2025-05-12 03:04:58,802 INFO datasources.FileSourceStrategy: Post-Scan Filters: isnotnull(continent#18)
2025-05-12 03:04:58,852 INFO memory.MemoryStore: Block broadcast_10 stored as values in memory (estimated size 487.8 KiB,
free 364.2 MiB)
```

（所有国家）每百万人死亡数量与年龄中位数关系

```
scala> spark.sql("select location,max(1000000*total_deaths/population),max(median_age) from data where continent is not nu
ll group by location").write.json("/dbcovid/result/million_deaths_with_age/")
```

（9）（所有国家）每百万人病例数量与人口密度关系

```
scala> spark.sql("select location,max(1000000*total_cases/population),max(population_density) from data where continent is
 not null group by location").write.json("/dbcovid/result/million_cases_with_density/")
```

（所有国家）每百万人死亡数量与人口密度关系

```
scala> spark.sql("select location,max(1000000*total_deaths/population),max(population_density) from data where continent i
s not null group by location").write.json("/dbcovid/result/million_deaths_with_density/")
```

（10）（所有国家）每百万人病例数量与期望寿命关系

```
scala> spark.sql("select location,max(1000000*total_cases/population),max(life_expectancy) from data where continent is no
t null group by location").write.json("/dbcovid/result/million_cases_with_lifecxp/")
```

（所有国家）每百万人死亡数量与期望寿命关系

```
scala> spark.sql("select location,max(1000000*total_deaths/population),max(life_expectancy) from data where continent is n
ot null group by location").write.json("/dbcovid/result/million_deaths_with_lifecxp/")
```

（11）（所有国家）每百万人病例数量与 HDI 关系

```
scala> spark.sql("select location,max(1000000*total_cases/population),max(human_development_index) from data where contine
nt is not null group by location").write.json("/dbcovid/result/million_cases_with_hdi/")
```

（所有国家）每百万人死亡数量与 HDI 关系

```
scala> spark.sql("select location,max(1000000*total_deaths/population),max(human_development_index) from data where contin
ent is not null group by location").write.json("/dbcovid/result/million_deaths_with_hdi/")
```

数据检查与导出

```
scala> bin/hdfs dfs -get /dbcovid/result ~/covid_data/
```

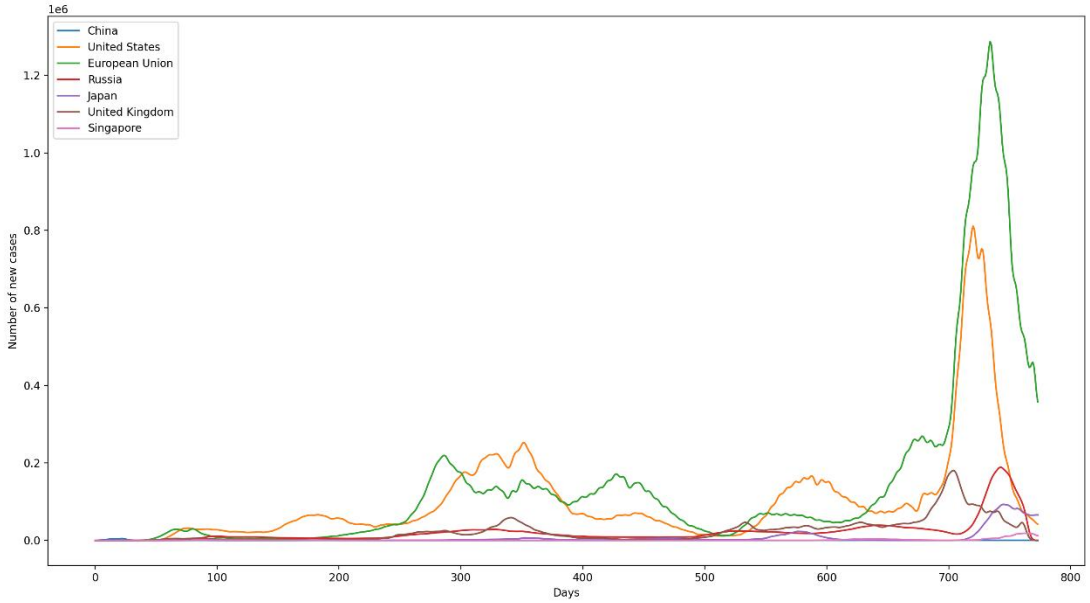
```
hadoop@master:/usr/local/hadoop$ bin/hdfs dfs -get /dbcovid/resu
lt ~/covid_data/
hadoop@master:/usr/local/hadoop$
```

4. 可视化分析

重点国家新增病例数量

```python
warnings.filterwarnings("ignore")

locs = ["China", "United States", "European Union", "Russia", "Japan", "United Kingdom", "Singapore"]

path_root = r"D:\spark\exam\covid\covid_data\result\new_cases"
all_data = []
valid_locs = []
for loc in locs:
    try:
        country_dir = os.path.join(path_root, loc)
        files = glob.glob(os.path.join(country_dir, "part-*.json"))
        if not files:
            print(f"警告: {loc} 目录下未找到part文件")
            continue

        country_df = pd.concat([pd.read_json(f, lines=True) for f in files])
        if country_df.empty:
            print(f"警告: {loc} 的数据为空")
            continue
        numeric_cols = country_df.select_dtypes(include=[np.number]).columns
        if len(numeric_cols) == 0:
            print(f"警告: {loc} 无数值列")
            continue

        tmp = country_df[numeric_cols[0]].values.astype(float)  # 转换为浮点数
        all_data.append(tmp)
        valid_locs.append(loc)

    except Exception as e:
        print(f"处理 {loc} 时出错: {str(e)}")
        continue
```

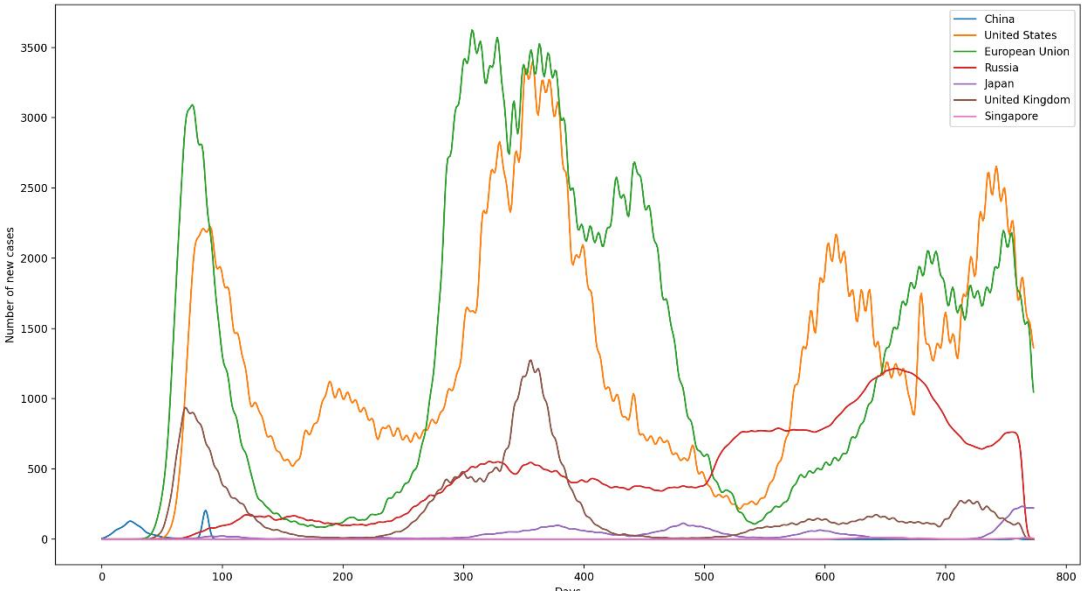
```python
if all_data:
    max_len = max(len(x) for x in all_data)
    padded_data = [np.pad(arr, (0, max_len - len(arr)),
                          mode='constant', constant_values=np.nan)
                   for arr in all_data]

    clean_data = []
    for arr in padded_data:
        mask = np.isnan(arr)
        if np.all(mask):
            arr[:] = 0
        else:
            arr[mask] = 0
        clean_data.append(arr)

    smoothed_data = [gaussian_filter1d(arr, sigma=2.5) for arr in clean_data]
    df = pd.DataFrame(np.array(smoothed_data).T, columns=valid_locs)

    plt.figure(figsize=(12, 8))
    plt.xlabel('Days')
    plt.ylabel('Number of new cases')
    sns.lineplot(data=df, dashes=False)
    plt.tight_layout()
    plt.show()
else:
    print("错误: 没有有效数据被加载")
```



重点国家新增死亡数量

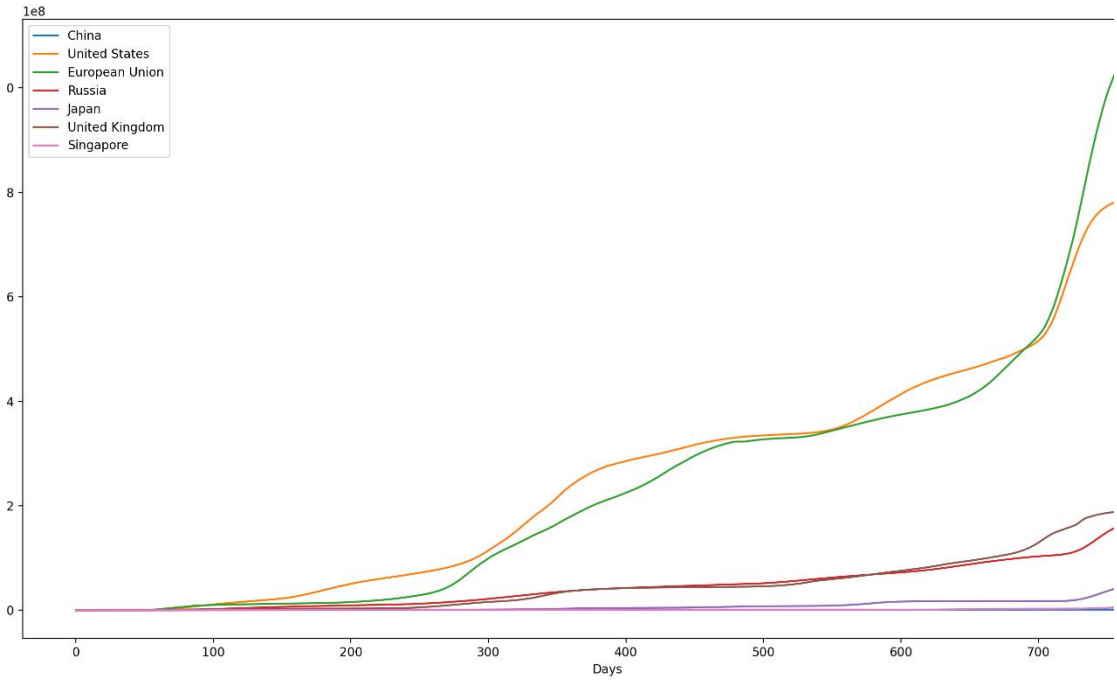
```python
locs = ["China", "United States", "European Union", "Russia", "Japan", "United Kingdom", "Singapore"]

path_root = r"D:\spark\exam\covid\covid_data\result\new_deaths"
all_data = []
valid_locs = []
for loc in locs:
    try:
        country_dir = os.path.join(path_root, loc)
        files = glob.glob(os.path.join(country_dir, "part-*.json"))
        if not files:
            print(f"警告: {loc} 目录下未找到part文件")
            continue

        country_df = pd.concat([pd.read_json(f, lines=True) for f in files])
        if country_df.empty:
            print(f"警告: {loc} 的数据为空")
            continue
        numeric_cols = country_df.select_dtypes(include=[np.number]).columns
        if len(numeric_cols) == 0:
            print(f"警告: {loc} 无数值列")
            continue

        tmp = country_df[numeric_cols[0]].values.astype(float)  # 转换为浮点数
        all_data.append(tmp)
        valid_locs.append(loc)

    except Exception as e:
        print(f"处理 {loc} 时出错: {str(e)}")
```

```python
if all_data:
    max_len = max(len(x) for x in all_data)
    padded_data = [np.pad(arr, (0, max_len - len(arr)),
                          mode='constant', constant_values=np.nan)
                   for arr in all_data]

    clean_data = []
    for arr in padded_data:
        mask = np.isnan(arr)
        if np.all(mask):
            arr[:] = 0
        else:
            arr[mask] = 0
        clean_data.append(arr)

    smoothed_data = [gaussian_filter1d(arr, sigma=2.5) for arr in clean_data]
    df = pd.DataFrame(np.array(smoothed_data).T, columns=valid_locs)

    plt.figure(figsize=(12, 8))
    plt.xlabel('Days')
    plt.ylabel('Number of new cases')
    sns.lineplot(data=df, dashes=False)
    plt.tight_layout()
    plt.show()
else:
    print("错误: 没有有效数据被加载")
```



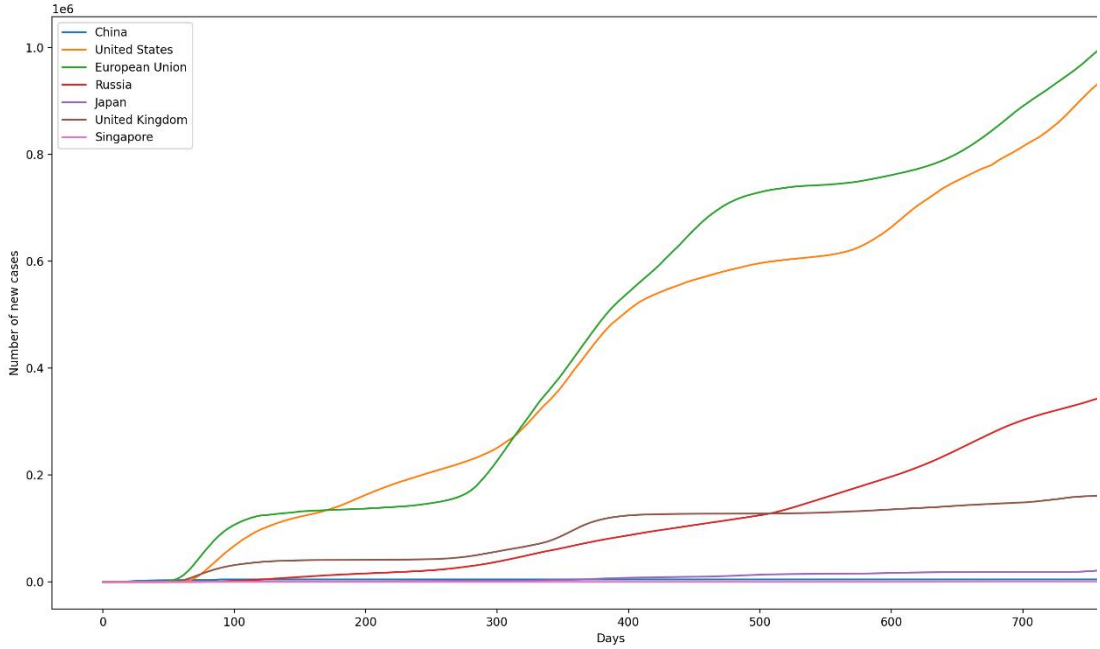
重点国家累计病例数量

```
locs = ["China", "United States", "European Union", "Russia", "Japan", "United Kingdom", "Singapore"]

path_root = r"D:\spark\exam\covid\covid_data\result\total_cases"
all_data = []
valid_locs = []
for loc in locs:
    try:
        country_dir = os.path.join(path_root, loc)
        files = glob.glob(os.path.join(country_dir, "part-*.json"))
        if not files:
            print(f"警告: {loc} 目录下未找到part文件")
            continue

        country_df = pd.concat([pd.read_json(f, lines=True) for f in files])
        if country_df.empty:
            print(f"警告: {loc} 的数据为空")
            continue
        numeric_cols = country_df.select_dtypes(include=[np.number]).columns
        if len(numeric_cols) == 0:
            print(f"警告: {loc} 无数值列")
            continue

        tmp = country_df[numeric_cols[0]].values.astype(float)  # 转换为浮点数
        all_data.append(tmp)
        valid_locs.append(loc)

    except Exception as e:
        print(f"处理 {loc} 时出错: {str(e)}")
        continue
```



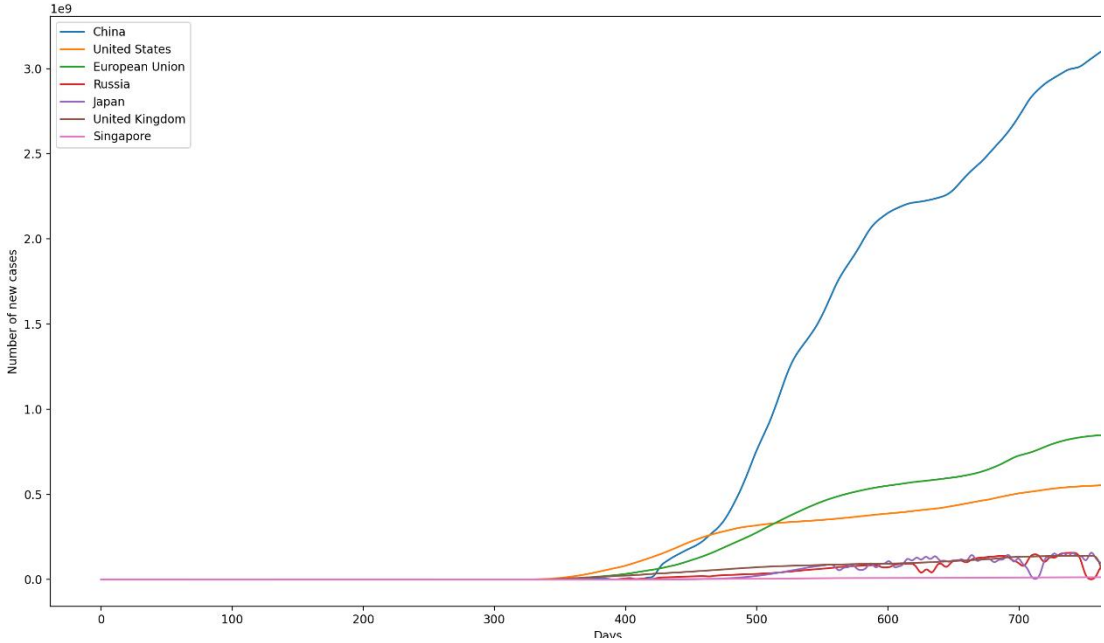
重点国家累计死亡数量

```
locs = ["China", "United States", "European Union", "Russia", "Japan", "United Kingdom", "Singapo

path_root = r"D:\spark\exam\covid\covid_data\result\total_deaths"
all_data = []
valid_locs = []
for loc in locs:
    try:
        country_dir = os.path.join(path_root, loc)
        files = glob.glob(os.path.join(country_dir, "part-*.json"))
        if not files:
            print(f"警告: {loc} 目录下未找到part文件")
            continue

        country_df = pd.concat([pd.read_json(f, lines=True) for f in files])
        if country_df.empty:
            print(f"警告: {loc} 的数据为空")
            continue
        numeric_cols = country_df.select_dtypes(include=[np.number]).columns
        if len(numeric_cols) == 0:
            print(f"警告: {loc} 无数值列")
            continue

        tmp = country_df[numeric_cols[0]].values.astype(float)  # 转换为浮点数
        all_data.append(tmp)
```



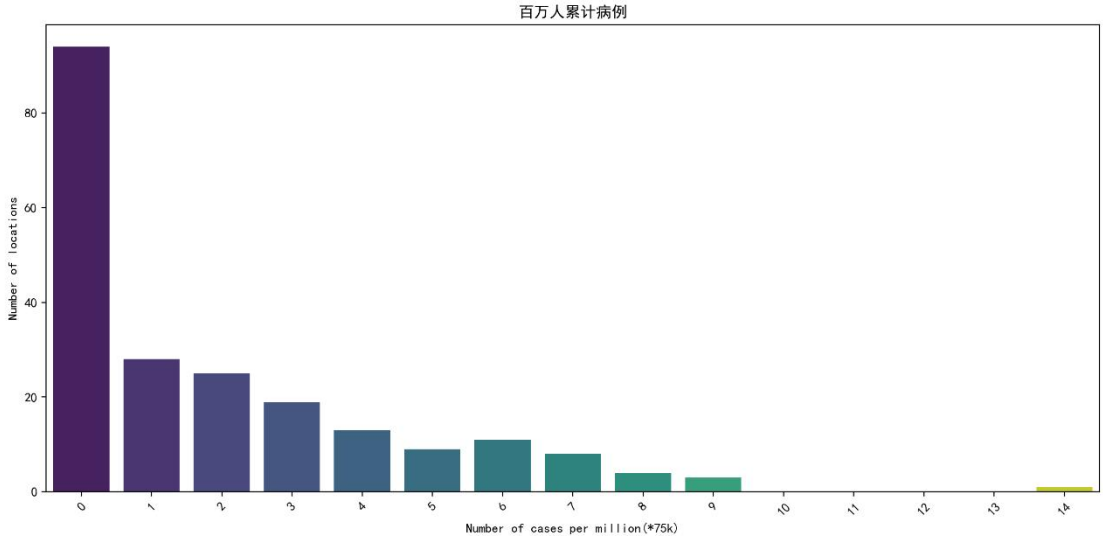
重点国家累计疫苗接种剂次

```
warnings.filterwarnings("ignore")

locs = ["China", "United States", "European Union", "Russia", "Japan", "United Kingdom", "Singapore"]

path_root = r"D:\spark\exam\covid\covid_data\result\total_vacc"
all_data = []
valid_locs = []
for loc in locs:
    try:
        country_dir = os.path.join(path_root, loc)
        files = glob.glob(os.path.join(country_dir, "part-*.json"))
        if not files:
            print(f"警告: {loc} 目录下未找到part文件")
            continue

        country_df = pd.concat([pd.read_json(f, lines=True) for f in files])
        if country_df.empty:
            print(f"警告: {loc} 的数据为空")
            continue
        numeric_cols = country_df.select_dtypes(include=[np.number]).columns
        if len(numeric_cols) == 0:
            print(f"警告: {loc} 无数值列")
            continue

        tmp = country_df[numeric_cols[0]].values.astype(float)  # 转换为浮点数
        all_data.append(tmp)
```



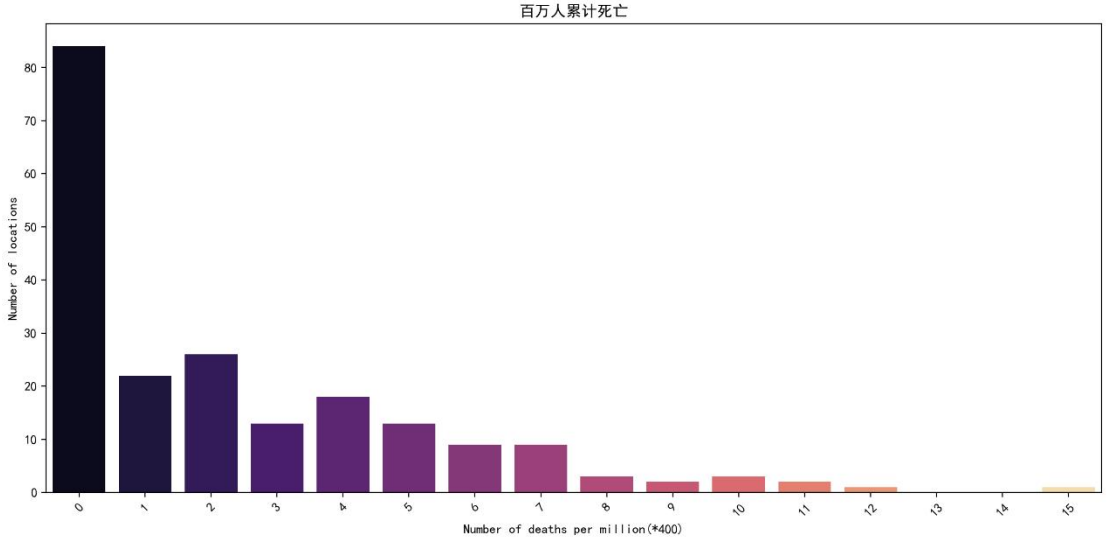
全部国家每百万人累计病例

```
plt.rcParams['axes.unicode_minus'] = False

def read_json_lines(file_path):  1 usage
    with open(file_path, 'r', encoding='utf-8') as f:
        data = [json.loads(line) for line in f]
    return pd.DataFrame(data)

def process_million_cases():  1 usage

    df = read_json_lines('D:/spark/exam/covid/covid_data/result/million_cases/million_cases.json')

    df.columns = ['location', 'cases_per_million']

    df = df.dropna()

    df = df.sort_values( by: 'cases_per_million', ascending=False)

    bins = [0, 50000, 100000, 150000, 200000, 250000, 300000, 350000, 400000, 450000, 500000, 550000, 600000, 650000,
            700000, float('inf')]
    labels = ['0', '1', '2', '3', '4', '5',
              '6', '7', '8', '9', '10', '11',
              '12', '13', '14']

    df['cases_group'] = pd.cut(df['cases_per_million'], bins=bins, labels=labels, right=False)

    count_data = df['cases_group'].value_counts().sort_index()

    plt.figure(figsize=(12, 6))
    sns.barplot(x=count_data.index, y=count_data.values, palette='viridis')
    plt.title('百万人累计病例')
```



全部国家每百万人累计死亡

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import json
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
def read_json_lines(file_path):  1 usage
    with open(file_path, 'r', encoding='utf-8') as f:
        data = [json.loads(line) for line in f]
    return pd.DataFrame(data)
def process_million_deaths():  1 usage
    df = read_json_lines('D:/spark/exam/covid/covid_data/result/million_deaths/million_deaths.json')
    df.columns = ['location', 'deaths_per_million']
    df = df.dropna()
    df = df.sort_values( by: 'deaths_per_million', ascending=False)

    bins = [0, 400, 800, 1200, 1600, 2000, 2400, 2800, 3200, 3600, 4000, 4400, 4800, 5200, 5600, 6000, float('inf')]
    labels = ['0','1', '2', '3', '4', '5', '6',
              '7', '8', '9', '10', '11',
              '12', '13', '14', '15']
    df['deaths_group'] = pd.cut(df['deaths_per_million'], bins=bins, labels=labels, right=False)
    count_data = df['deaths_group'].value_counts().sort_index()
    plt.figure(figsize=(12, 6))
    sns.barplot(x=count_data.index, y=count_data.values, palette='magma')
    plt.title('百万人累计死亡')
    plt.xlabel('Number of deaths per million(*400)')
    plt.ylabel('Number of locations')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

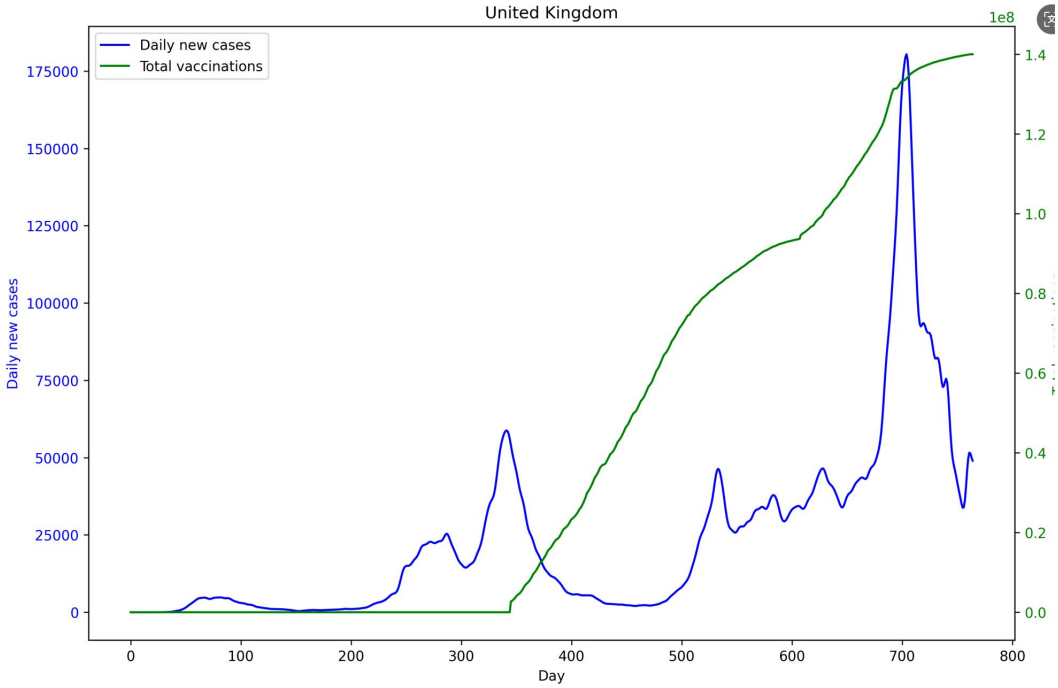


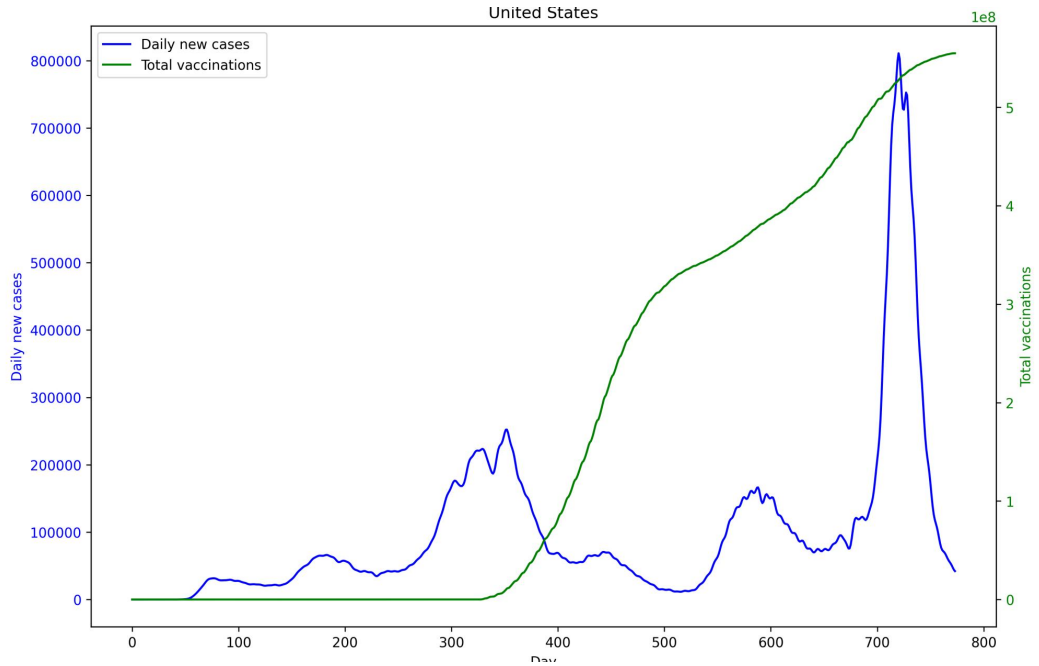
相关性分析

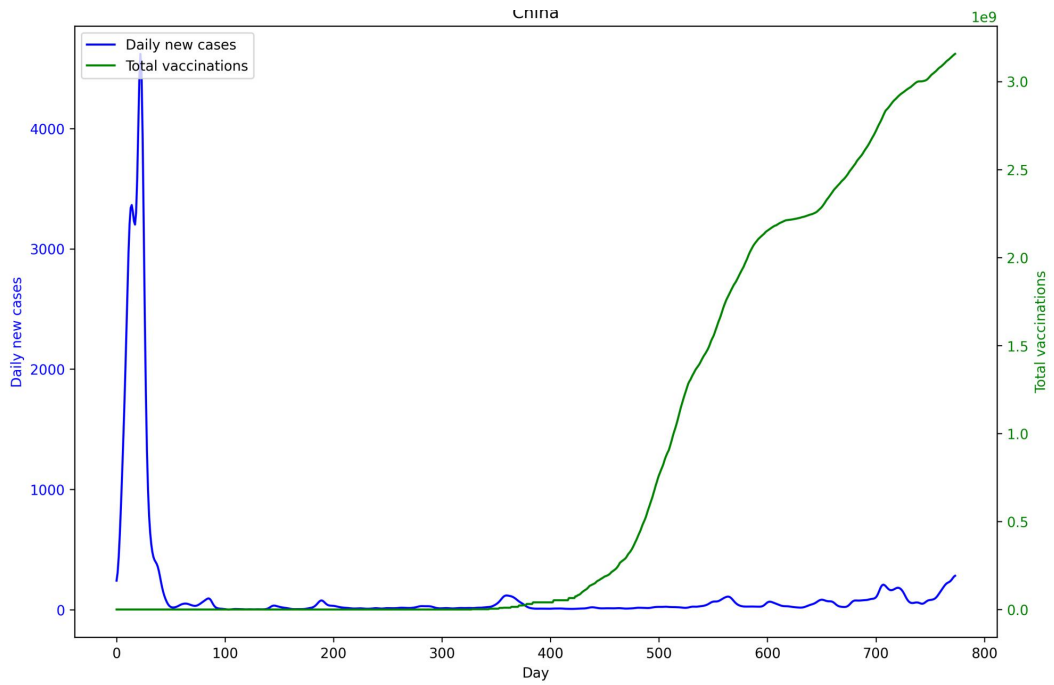
重点国家新增病例数量与疫苗剂次关系

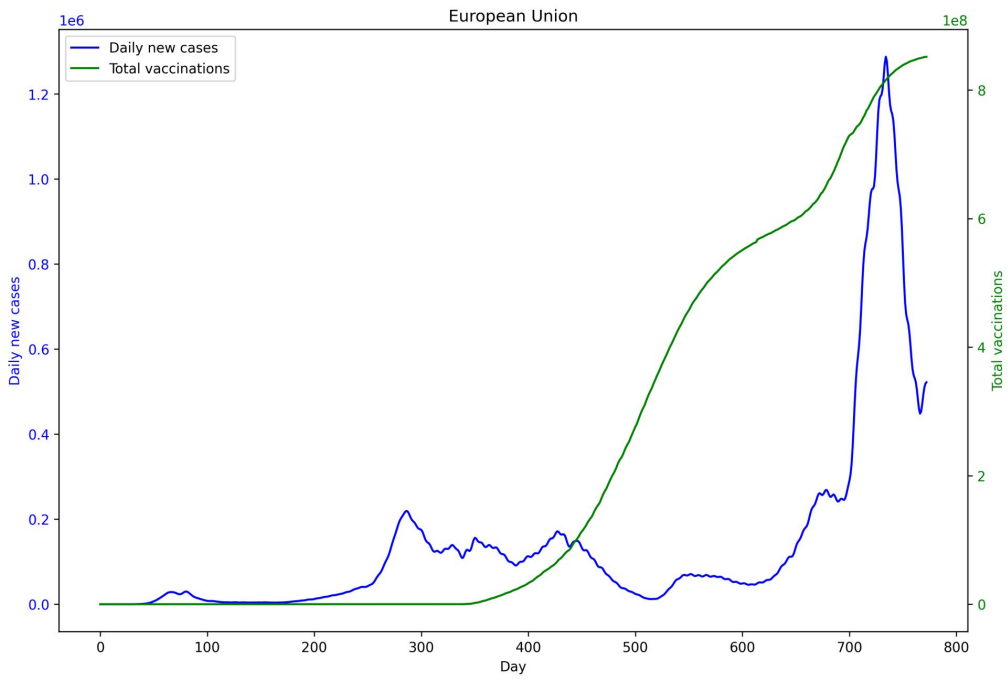
```
            print(f"警告: {loc} 数据列不足2列")
            continue
        data = country_df[numeric_cols[:2]].values.T  # 只取前两列
        for col in data:
            if np.isnan(col[0]):
                col[0] = 0
            for i in range(len(col) - 1):
                if np.isnan(col[i + 1]):
                    col[i + 1] = col[i]
        data[0] = gaussian_filter1d(data[0], sigma=2.5)
        plt.figure(figsize=(12, 8))
        plt.title(loc)
        ax1 = plt.gca()
        ax1.plot( *args: data[0], 'b-', label='Daily new cases')
        ax1.set_xlabel('Day')
        ax1.set_ylabel('Daily new cases', color='b')
        ax1.tick_params(axis='y', labelcolor='b')
        ax2 = ax1.twinx()
        ax2.plot(data[1], 'g-', label='Total vaccinations')
        ax2.set_ylabel('Total vaccinations', color='g')
        ax2.tick_params(axis='y', labelcolor='g')
        lines1, labels1 = ax1.get_legend_handles_labels()
        lines2, labels2 = ax2.get_legend_handles_labels()
        ax1.legend(lines1 + lines2, labels1 + labels2, loc='upper left')
        plt.savefig( *args: f'新增与疫苗关系_{loc}.png', bbox_inches='tight', dpi=300)
        plt.close()
    except Exception as e:
        print(f"处理 {loc} 时出错: {str(e)}")
        continue
print("处理完成")
```

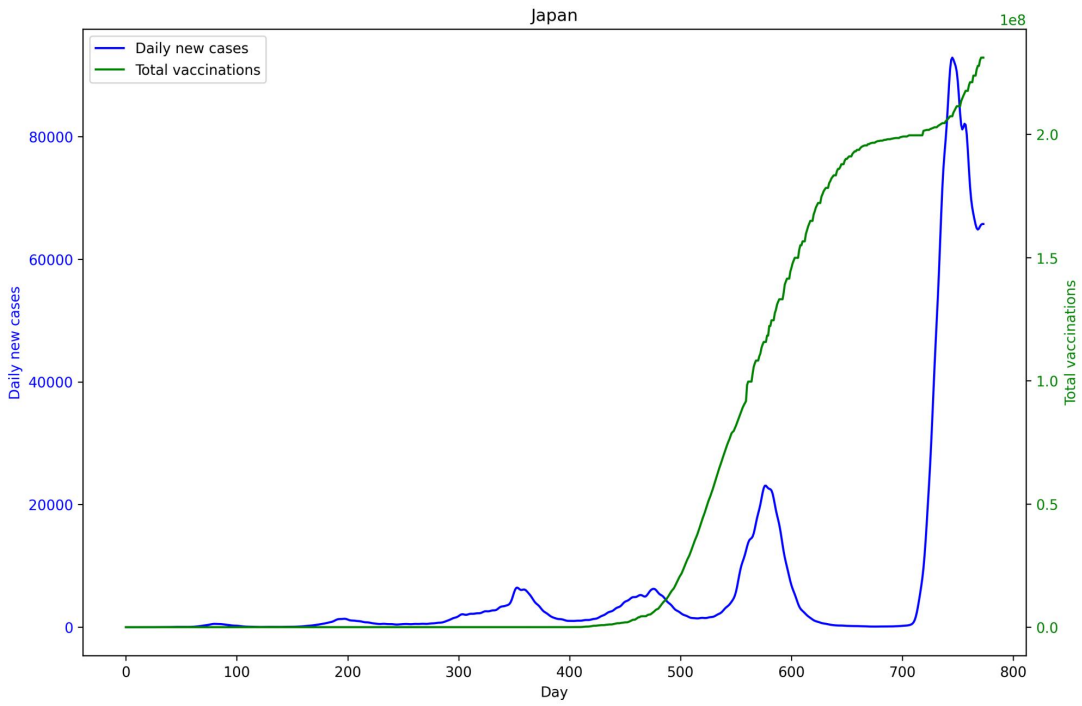
新增与疫苗关系_China.png	2025/5/11 17:25	PNG 图
新增与疫苗关系_European Union.png	2025/5/11 17:25	PNG 图
新增与疫苗关系_Japan.png	2025/5/11 17:25	PNG 图
新增与疫苗关系_Russia.png	2025/5/11 17:25	PNG 图
新增与疫苗关系_Singapore.png	2025/5/11 17:25	PNG 图
新增与疫苗关系_United Kingdom.png	2025/5/11 17:25	PNG 图
新增与疫苗关系_United States.png	2025/5/11 17:25	PNG 图

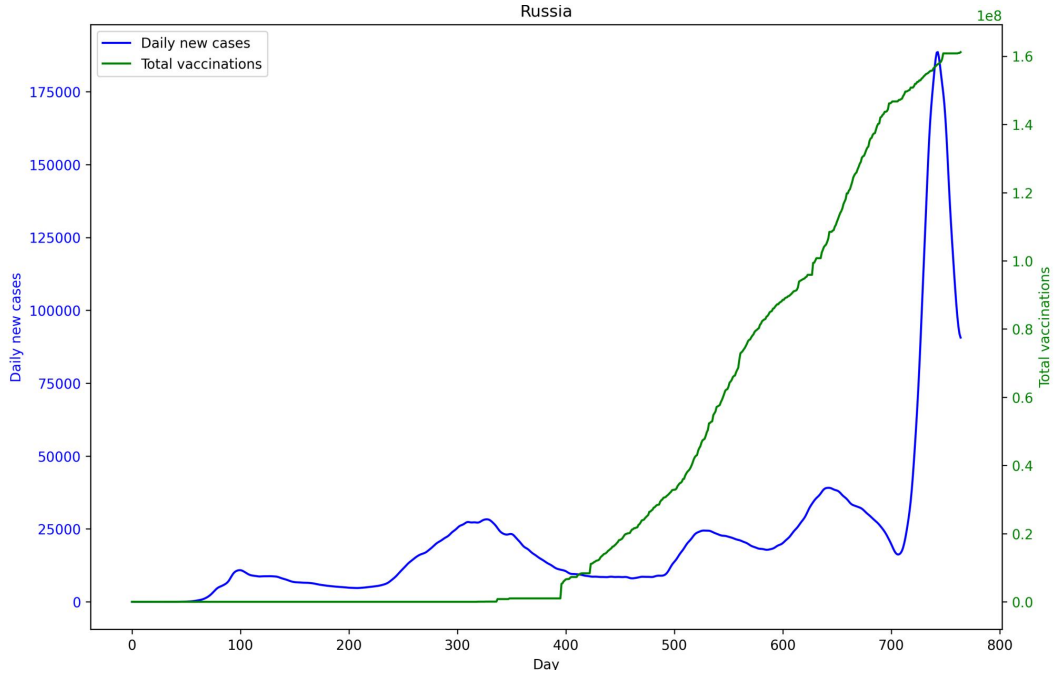


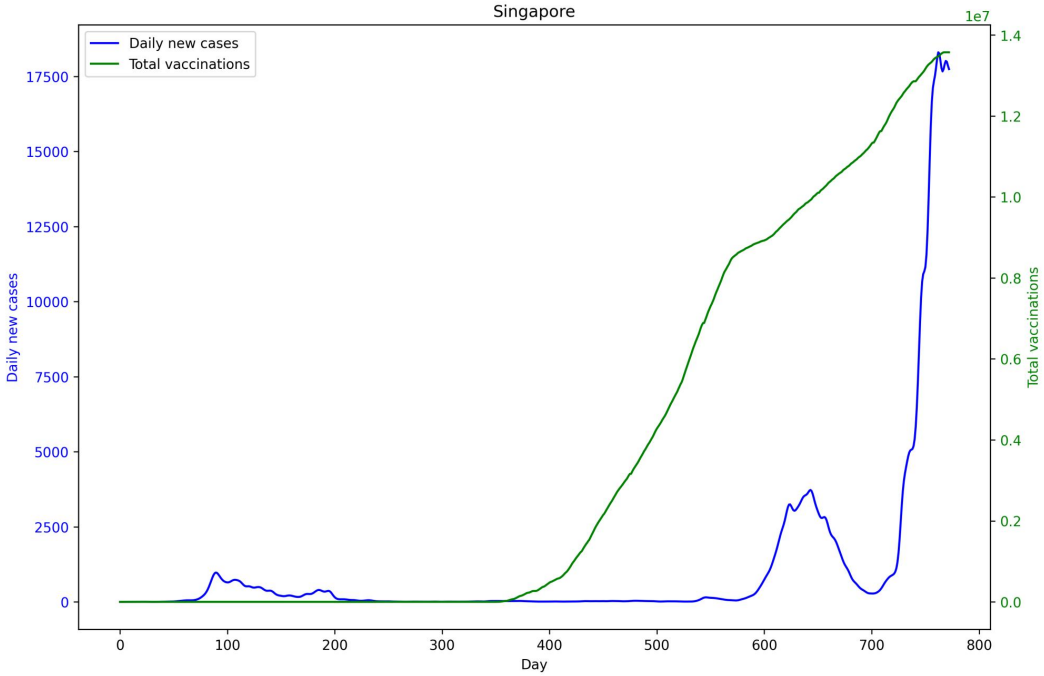

United States
1e8
Daily new cases
Total vaccinations
Daily new cases
Total vaccinations
0
100000
200000
300000
400000
500000
600000
700000
800000
0
1
2
3
4
5
0
100
200
300
400
500
600
700
800


China
1e9
Daily new cases
Total vaccinations
Daily new cases
Total vaccinations
0
1000
2000
3000
4000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
100
200
300
400
500
600
700
800
Day

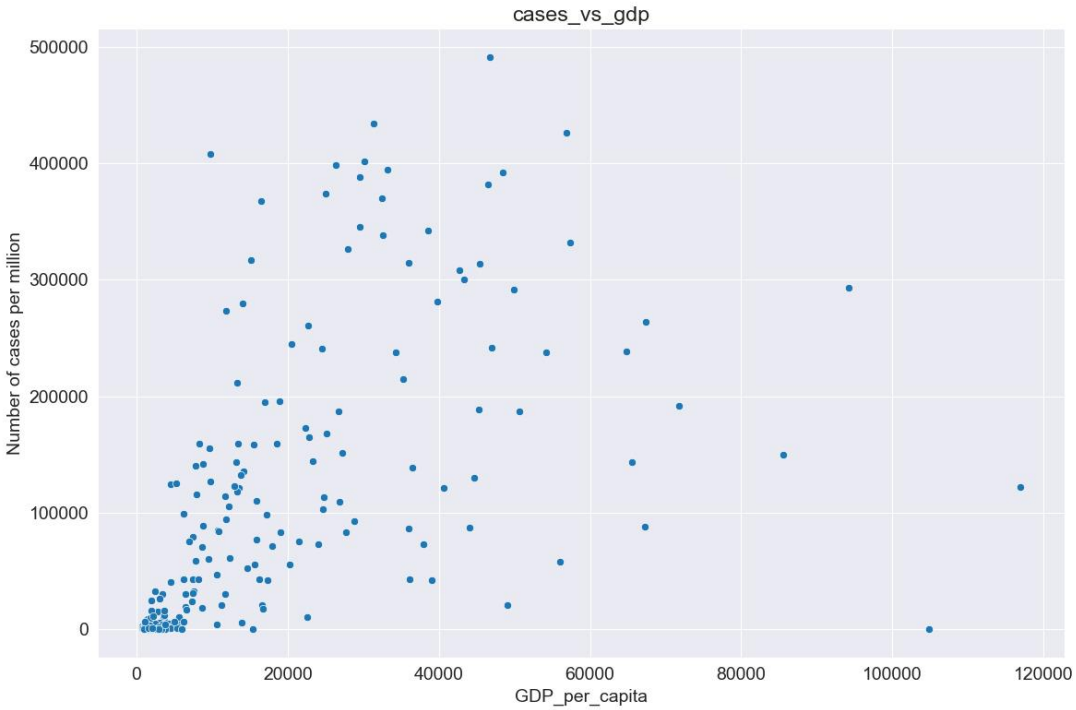

European Union
1e6
1e8
Daily new cases
Total vaccinations
Daily new cases
Total vaccinations
Day
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
2
4
6
8
0
100
200
300
400
500
600
700
800


Japan
1e8
Daily new cases
Total vaccinations
Daily new cases
Total vaccinations
Day
0
20000
40000
60000
80000
0.0
0.5
1.0
1.5
2.0
0
100
200
300
400
500
600
700
800

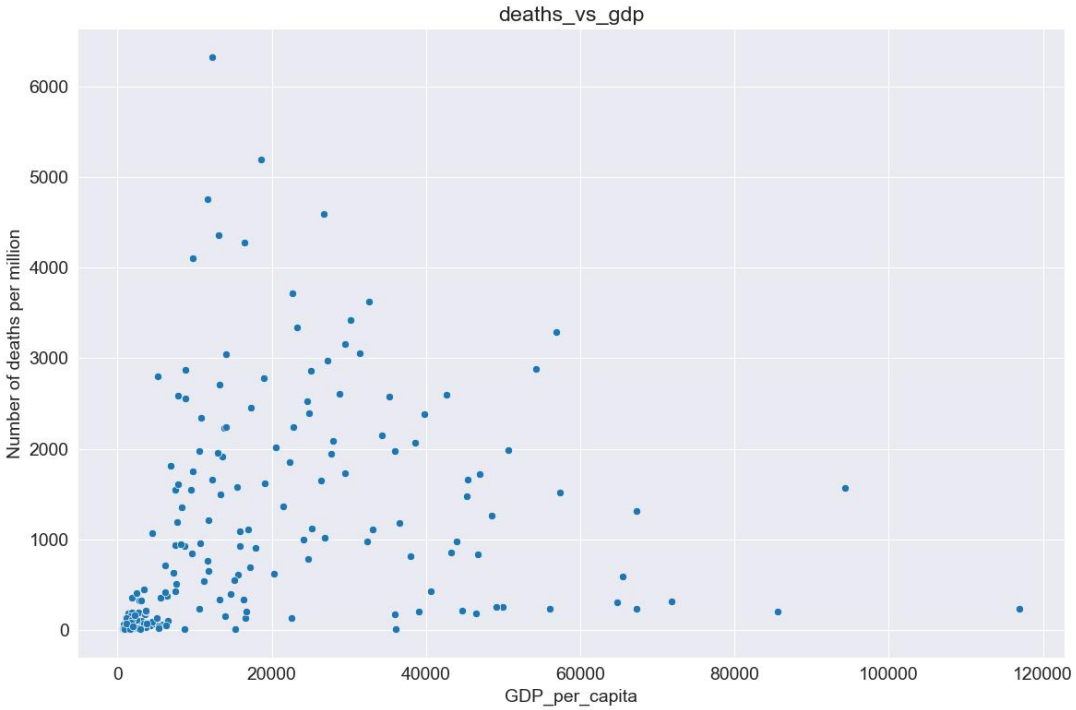

Russia
Daily new cases
Total vaccinations
1e8
Daily new cases
Total vaccinations
Day
0
25000
50000
75000
100000
125000
150000
175000
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0
100
200
300
400
500
600
700
800


Singapore
Daily new cases
Total vaccinations
1e7
Daily new cases
Total vaccinations
Day
0
2500
5000
7500
10000
12500
15000
17500
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0
100
200
300
400
500
600
700
800

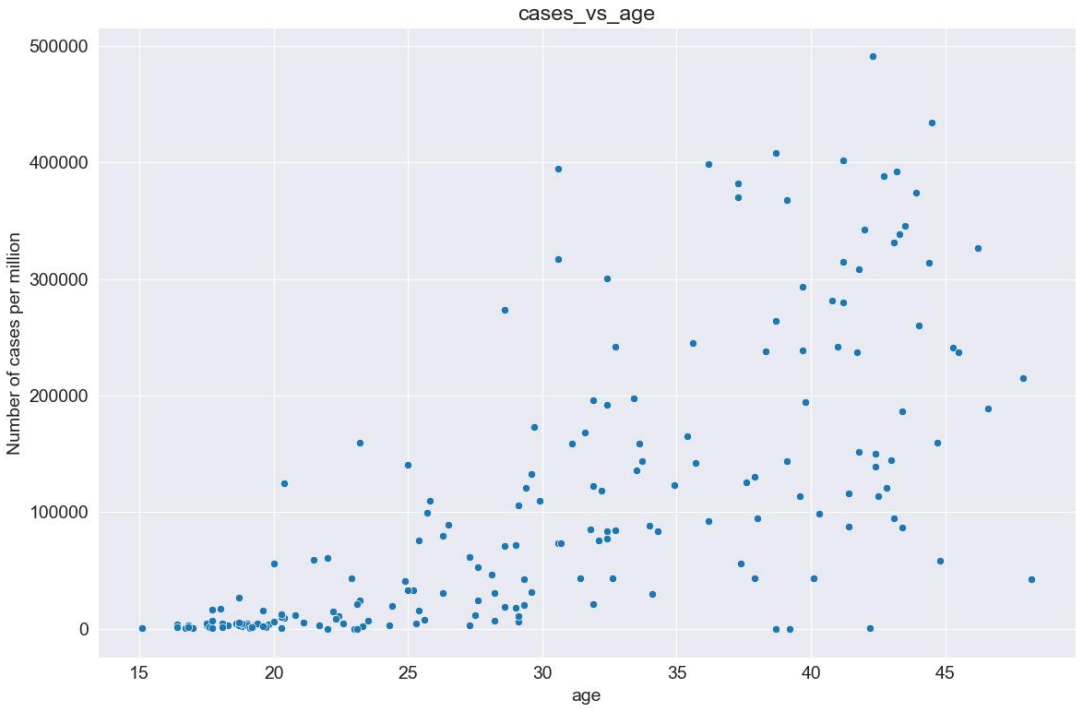
全部国家每百万人累计病例与人均 GDP 关系



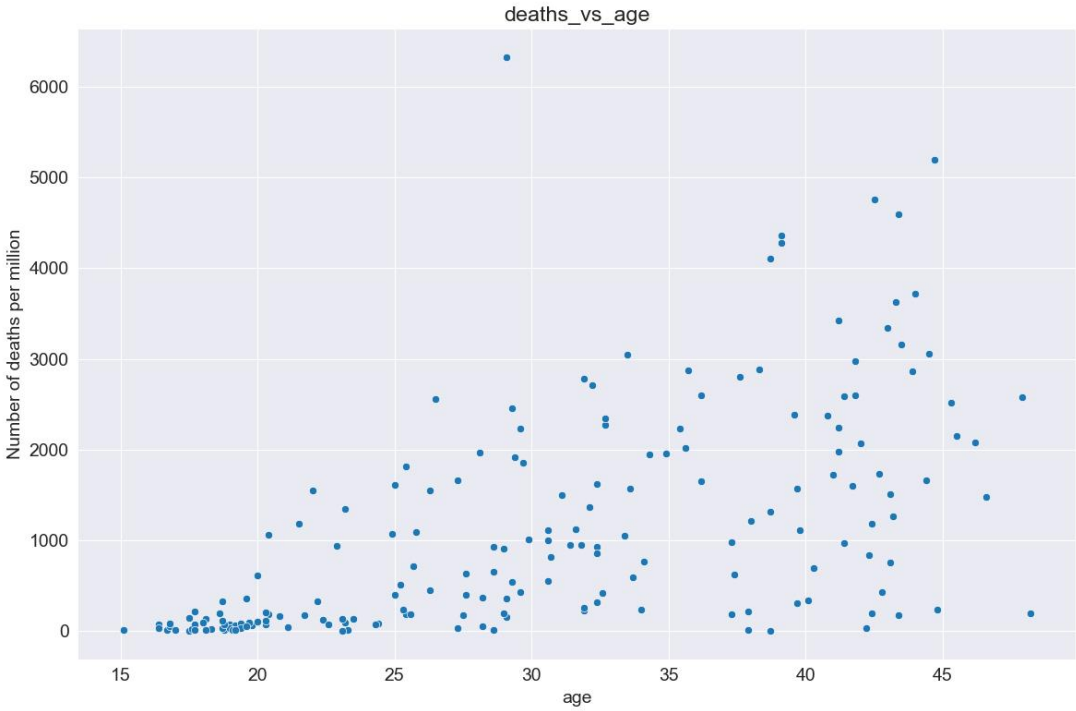
全部国家每百万人累计死亡与人均 GDP 关系



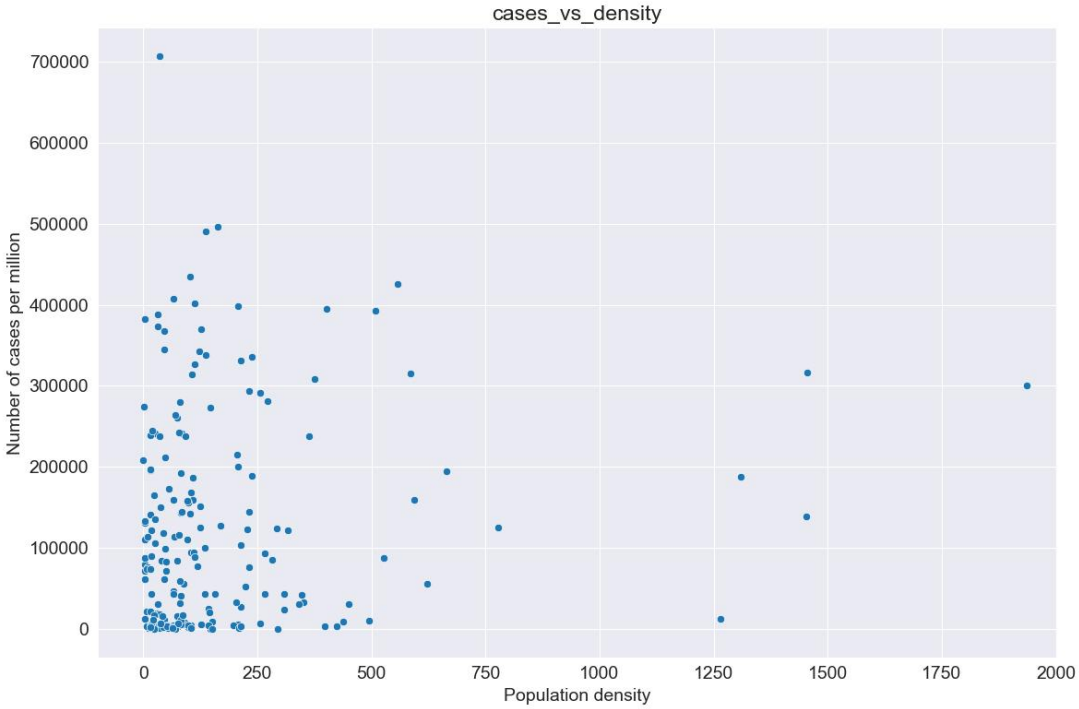
全部国家每百万人累计病例与年龄中位数关系



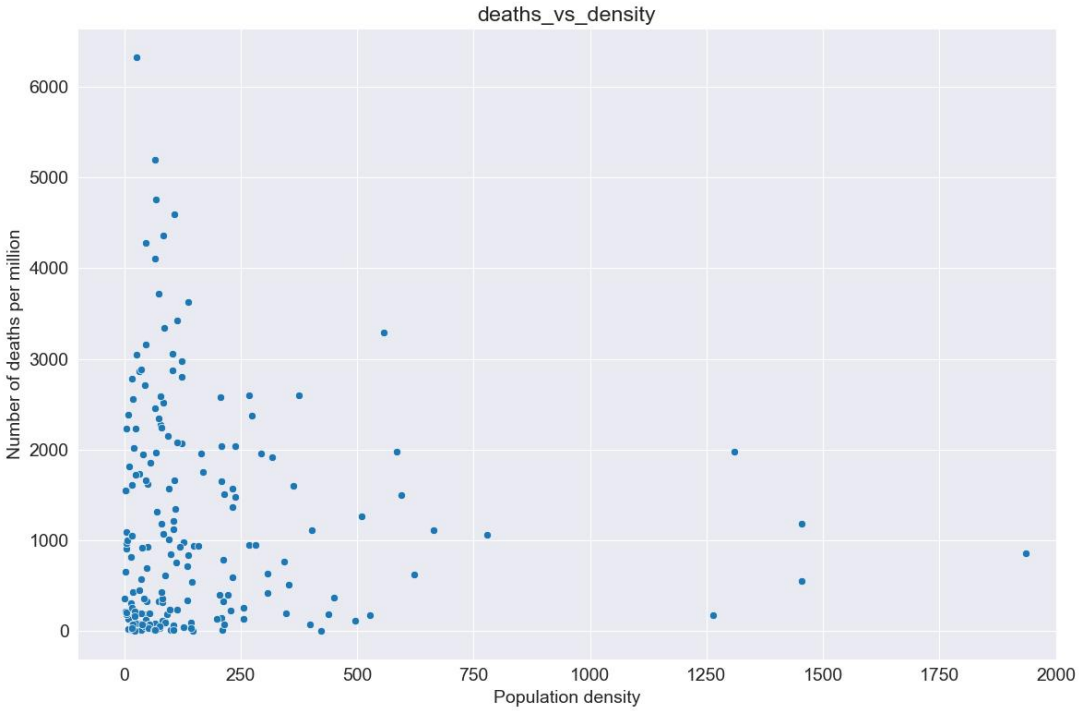
全部国家每百万人累计死亡与年龄中位数关系



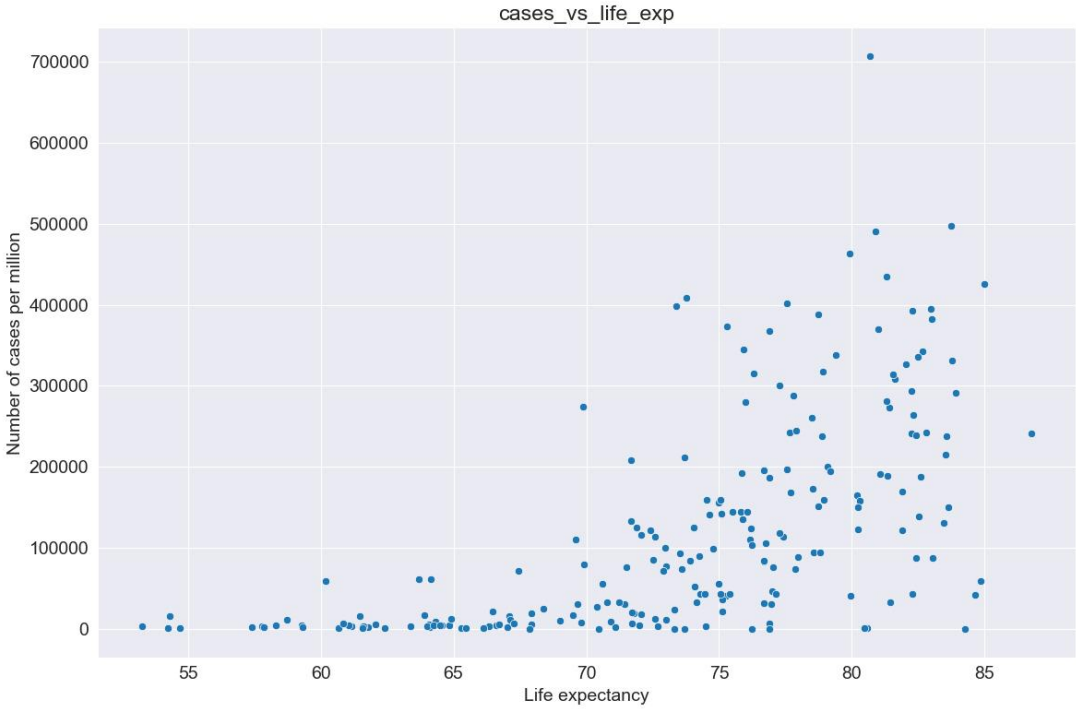
全部国家每百万人累计病例与人口密度关系



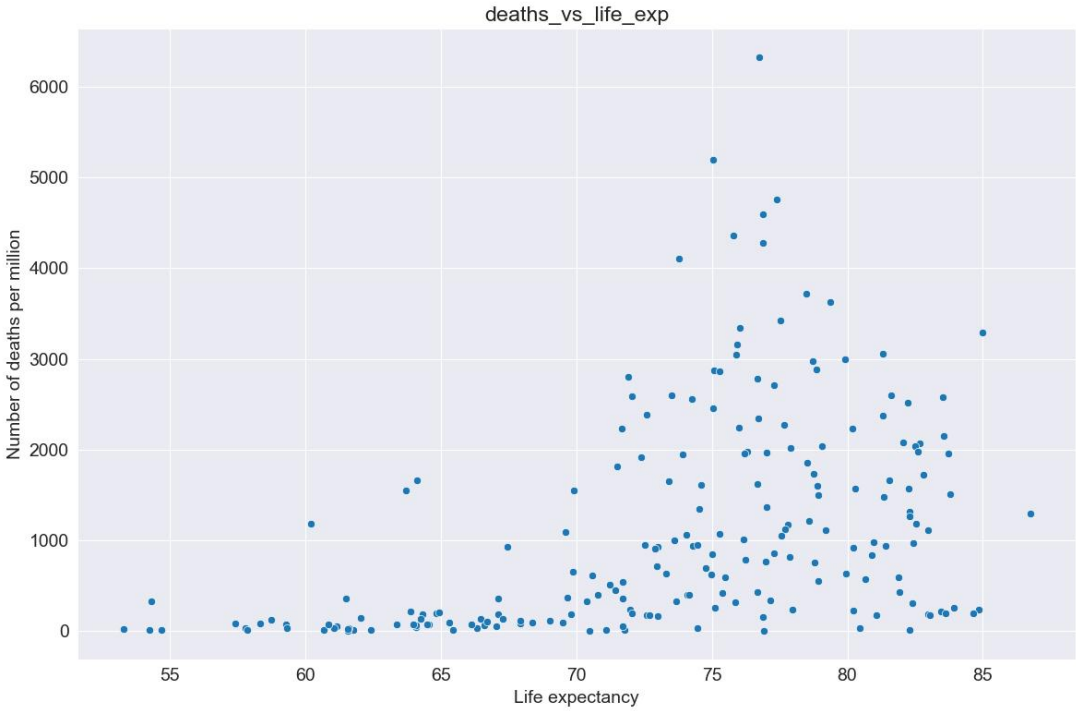
全部国家每百万人累计死亡与人口密度关系



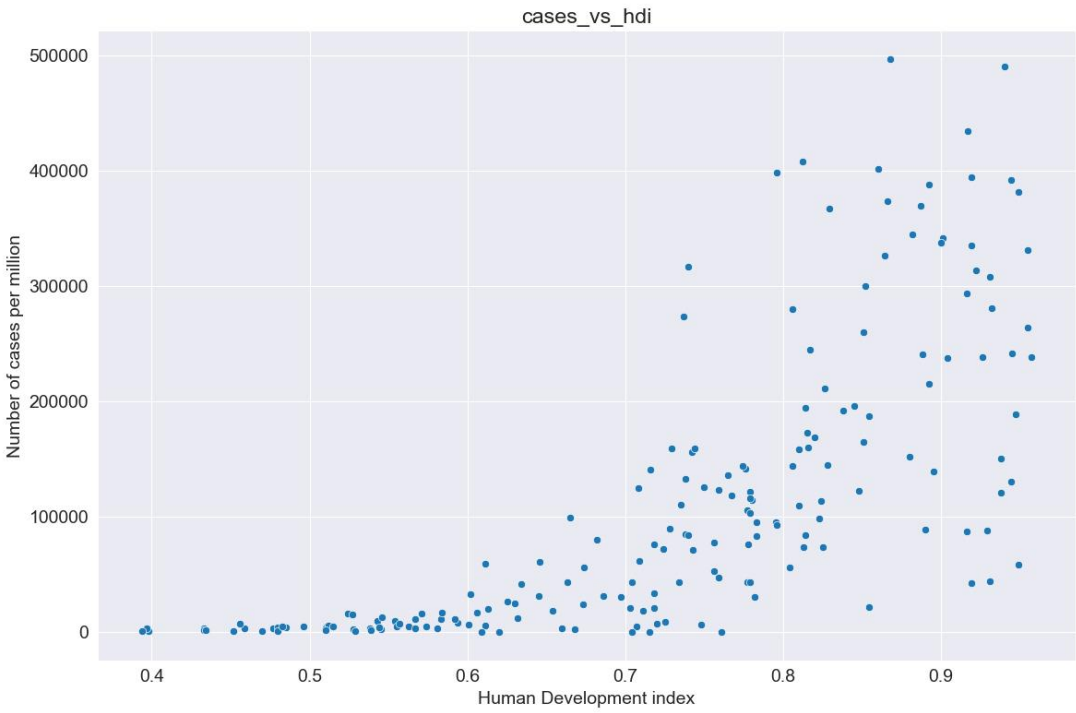
全部国家每百万人累计病例与期望寿命关系



全部国家每百万人累计死亡与期望寿命关系



全部国家每百万人累计病例与 HDI 关系



全部国家每百万人累计死亡与 HDI 关系