疫情可视化案例

1. 世界各国家现存确诊人数地图

将国家或地区的数值信息映射到地图上，通过颜色变化来表示数值的大小或范围。

颜色地图适合带有地理位置信息的数据的展现，将颜色和地图相结合，直观显示数据的地理分布，通过颜色深浅容易判断数值的大小。

In [1]:
```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

world_data = pd.read_csv('data/today_world_2020_04_04.csv')
```

In [2]:
```
import pyecharts          #可以使用pip install pyecharts安装，官方文档:https://pyecharts.org/#/zh-cn/intro
# 调整配置项
import pyecharts.options as opts
# Map类用于绘制地图
from pyecharts.charts import Map
```

In [3]:
```
world_data['today_storeConfirm'] = world_data['total_confirm'] - world_data['total_heal'] - world_data['total_dead']
#利用各国家的累计确诊人数减去累计治愈和累计死亡人数得到现存确诊人数，作为新的一列特征添加到数据中；
```

In [4]:
```
world_data.head()
```

Out[4]:

	id	lastUpdateTime	name	total_confirm	total_suspect	total_heal	total_dead	total_severe	total_input	today_confirm	today_suspect	today_heal	today_dead	today_severe	today_storeConfirm	today_input
0	9577772	2020/4/4	突尼斯	495	0	5	18	0	NaN	40.0	0.0	2.0	4.0	0.0	472	NaN
1	9507896	2020/4/4	塞尔维亚	1476	0	0	89	0	NaN	NaN	NaN	NaN	NaN	NaN	1387	NaN
2	0	2020/4/4	中国	82899	0	76992	3335	0	888.0	97.0	0.0	207.0	4.0	0.0	2572	18.0
3	1	2020/4/4	日本	4072	0	1133	89	0	NaN	209.0	NaN	0.0	0.0	NaN	2850	NaN
4	2	2020/4/4	泰国	2067	0	612	20	0	NaN	89.0	0.0	31.0	1.0	0.0	1435	NaN

利用Map类绘制世界地图时，需要输入各国家的英文名称和对应的数值，但我们的数据中国家的名称是中文，所以我们首先要将中文名称转换为英文名称：

In [5]:
```
#载入国家中英文对照表，使用Series类的replace方法将各国家对应的中文名称替换成英文名称；
contry_name = pd.read_csv('data/county_china_english.csv', encoding='GB2312')
contry_name.head()
```

Out[5]:

	英文	中文
0	Afghanistan	阿富汗
1	Aland Islands	奥兰群岛
2	Albania	阿尔巴尼亚
3	Algeria	阿尔及利亚
4	American Samoa	美属萨摩亚

In [6]:
```
world_data['eg_name'] = world_data['name'].replace(contry_name['中文'].values ,contry_name['英文'].values)
world_data['eg_name'].head()
```

Out[6]:
```
0     Tunisia
1      Serbia
2       China
3       Japan
4    Thailand
Name: eg_name, dtype: object
```

In [7]:
```
#提取出我们需要的数据，保存成一个嵌套列表的形式
heatmap_data = world_data[['eg_name','today_storeConfirm']].values.tolist()
heatmap_data[:10]
```

Out[7]:
```
[['Tunisia', 472],
 ['Serbia', 1387],
 ['China', 2572],
 ['Japan', 2850],
 ['Thailand', 1435],
 ['Singapore', 843],
 ['Korea', 3961],
 ['Australia', 4817],
 ['Germany', 69721],
 ['United States', 261454]]
```

In [8]:
```
#开始绘图，首先初始化类对象Map，并调用add方法添加绘图基本配置；
map_ = Map().add(series_name = "现存确诊人数", # 设置提示框标签
                 data_pair = heatmap_data, # 输入数据
                 maptype = "world", # 设置地图类型为世界地图
                 is_map_symbol_show = False # 不显示标记点
                 )
```

In [9]:
```
# 设置系列配置项
map_.set_series_opts(label_opts=opts.LabelOpts(is_show=False))  # 不显示国家（标签）名称
```

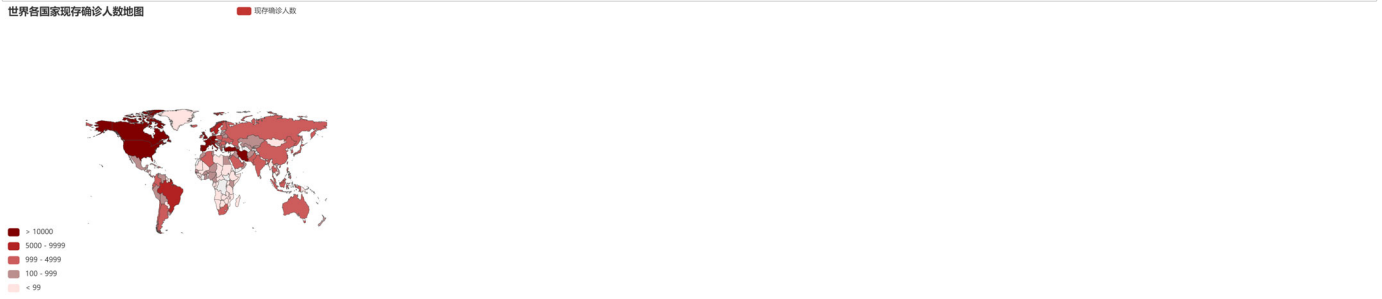
Out[9]:
```
<pyecharts.charts.basic_charts.map.Map at 0x1e64c1bb0a0>
```

In [10]:
```
# 设置全局配置项
map_.set_global_opts(title_opts = opts.TitleOpts(title="世界各国家现存确诊人数地图"), # 设置图标题
                     # 设置视觉映射配置项
                     visualmap_opts = opts.VisualMapOpts(pieces=[ # 自定义分组的分点和颜色
                                                                 {"min": 10000,"color":"#800000"}, # 栗色
                                                                 {"min": 5000, "max": 9999, "color":"#B22222"}, # 耐火砖
                                                                 {"min": 999, "max": 4999,"color":"#CD5C5C"}, # 印度红
                                                                 {"min": 100, "max": 999, "color":"#BC8F8F"}, # 玫瑰棕色
                                                                 {"max": 99, "color":"#FFE4E1"}, # 薄雾玫瑰
                                                                ],
                     is_piecewise = True))  # 显示分段式图例
```

Out[10]:
```
<pyecharts.charts.basic_charts.map.Map at 0x1e64c1bb0a0>
```

In [11]:
```
# 在notebook中进行渲染
map_.render_notebook()
```

Out[11]:



2. 世界国家累计死亡人数玫瑰图

玫瑰图是一种二维极坐标统计图。玫瑰图与饼状图类似，饼状图各个扇形的半径相同，角度不同，角度表示每一部分占比的大小；

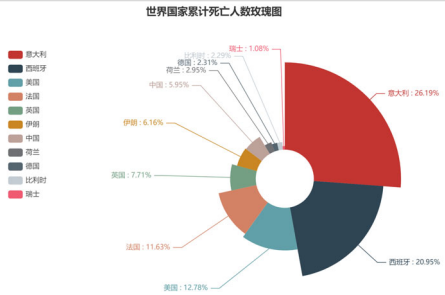
玫瑰图各个扇形的半径和角度都不同，角度依然表示每一部分的占比大小，半径表示每一部分的数值大小。

In [12]:
```
#首先筛选出累计死亡人数超过500人的世界国家，并按人数进行降序排序；
need_data = world_data[['name','total_dead']][world_data['total_dead'] >500]
rank = need_data[['name','total_dead']].sort_values(by='total_dead',ascending=False).values
```

In [13]:
```
#接着导入Pie类并添加绘图的基本配置；
from pyecharts.charts import Pie

pie = Pie().add("累计死亡人数", # 添加提示框标签
                rank, # 输入数据
                radius = ["20%", "80%"],  # 设置内半径和外半径
                center = ["60%", "60%"],  # 设置圆心位置
                rosetype = "radius")  # 玫瑰图模式，通过半径区分数值大小，角度大小表示占比
```

In [14]:
```
pie.set_global_opts(title_opts = opts.TitleOpts(title="世界国家累计死亡人数玫瑰图",  # 设置图标题
                                                pos_right = '40%'),  # 图标题的位置
                    legend_opts = opts.LegendOpts( # 设置图例
                                                  orient='vertical', # 垂直放置图例
                                                  pos_right="85%", # 设置图例位置
                                                  pos_top="15%"))

pie.set_series_opts(label_opts = opts.LabelOpts(formatter="{b} : {d}%")) # 设置标签文字形式为（国家：占比（%））

# 在notebook中进行渲染
pie.render_notebook()
```

Out[14]:



3. 3月世界国家累计确诊人数动态条形图

条形图由一些长度不等的横向长方形组成，以长方形的长度来表示数据，易于比较各组数据之间的差别。

In [15]:
```
#首先挑选出疫情最为严重的10个国家，并筛选出这些国家的历史疫情数据：
country_list = ['美国', '意大利', '中国', '西班牙', '德国', '伊朗', '法国', '英国', '瑞士','比利时']
alltime_data = pd.read_csv('data/alltime_world_2020_04_04.csv')
need_data = alltime_data[alltime_data['name'].isin(country_list)]
```

In [16]:
```
#使用datetime模块生成时间数据，构造时间列表：
from datetime import datetime,timedelta
time_list = [(datetime(2020, 3, 1) + timedelta(i)).strftime('%Y-%m-%d') for i in range(31)]
```

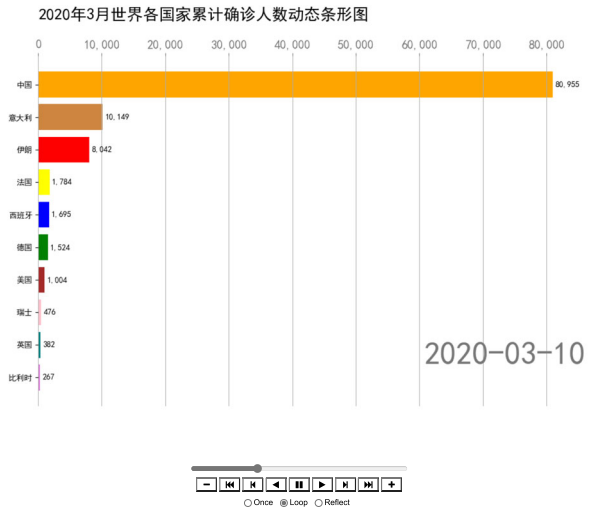
In [17]:
```
#载入Matplotlib库，并设置正常显示中文字体：
import matplotlib.pyplot as plt
%matplotlib inline

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['figure.dpi'] = 100
```

In [18]:
```
#为每个国家设置一种颜色：
color_list = ['brown','peru','orange','blue','green',
              'red','yellow','teal','pink','orchid']
country_color = pd.DataFrame()
country_color['country'] = country_list
country_color['color'] = color_list
```

In [19]:
```
#定义绘图函数：
import matplotlib.ticker as ticker

def barh_draw(day):

    # 提取每一天的数据
    draw_data = need_data[need_data['date']==day][['name','total_confirm']].sort_values(by='total_confirm',ascending=True)

    # 清空当前的绘图
    ax.clear()

    # 绘制条形图
    ax.barh(draw_data['name'],draw_data['total_confirm'], color=[country_color[country_color['country']==i]['color'].values[0] for i in draw_data['name']])

    # 数值标签的间距
    dx = draw_data['total_confirm'].max()/200

    # 添加数值标签
    for j, (name, value) in enumerate(zip(draw_data['name'], draw_data['total_confirm'])):

        ax.text(value+dx, j, f'{value:,.0f}', size=10, ha='left', va='center')

    # 添加日期标签
    ax.text(draw_data['total_confirm'].max()*0.75, 0.4, day, color='#777777',size=40, ha='left')

    # 设置刻度标签的格式
    ax.xaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))

    # 设置刻度的位置
    ax.xaxis.set_ticks_position('top')

    # 设置刻度标签的颜色和大小
    ax.tick_params(axis='x',colors='#777777', labelsize=15)

    # 添加网格线
    ax.grid(which='major',axis='x',linestyle='-')

    # 添加图标题
    ax.text(0, 11, '2020年3月世界各国家累计确诊人数动态条形图',size=20, ha='left')

    # 去除图边框
    plt.box(False)

    # 关闭绘图框
    plt.close()
```

In [28]:
```
# 动态绘图
fig, ax = plt.subplots(figsize=(12, 8))

import matplotlib.animation as animation
from IPython.display import HTML

animator = animation.FuncAnimation(fig, barh_draw, frames=time_list, interval=200)
HTML(animator.to_jshtml())
```

Out[28]:



4.3月美国单日新增确诊人数与股票指数涨跌幅折线图

折线图可以显示随时间而变化的连续数据，因此非常适合显示在相等时间间隔下数据的趋势。

通过折线图可以观察美国疫情对美国股票乃至世界其它股票的影响。

In [21]:
```
#首先我们读取从API上获取的股票指数数据，由于周末股市不开盘，所以只有22天的数据；
stock = pd.read_csv('./data/stockindex.csv',encoding='GB2312')
stock
```

Out[21]:

	日期	NASDAQ	SSEC	N225
0	2020-03-02	4.49	3.15	0.95
1	2020-03-03	-2.99	0.74	-1.22
2	2020-03-04	3.85	0.63	0.08
3	2020-03-05	-3.10	1.99	1.09
4	2020-03-06	-1.86	-1.21	-2.72
5	2020-03-09	-7.29	-3.01	-5.07
6	2020-03-10	4.95	1.82	0.85
7	2020-03-11	-4.70	-0.94	-2.27
8	2020-03-12	-9.43	-1.52	-4.41
9	2020-03-13	9.35	-1.23	-6.08
10	2020-03-16	-12.32	-3.40	-2.46
11	2020-03-17	6.23	-0.34	0.06
12	2020-03-18	-4.70	-1.83	-1.68
13	2020-03-19	2.30	-0.98	-1.04
14	2020-03-20	-3.79	1.61	0.19
15	2020-03-23	-0.27	-3.11	2.02
16	2020-03-24	8.12	2.34	7.13
17	2020-03-25	-0.45	2.17	8.04
18	2020-03-26	5.60	-0.60	-4.51
19	2020-03-27	-3.79	0.26	3.88
20	2020-03-30	3.62	-0.90	-1.57
21	2020-03-31	-0.95	0.11	-0.88

In [22]:
```
#筛选出与股票开盘日期对应的美国单日新增确诊人数；
alltime_us = alltime_data[alltime_data['name'] == '美国']
use_data = alltime_us[alltime_data['date'].isin(stock['日期'].values)][['date','today_confirm']]
```

In [23]:
```
from pyecharts.charts import Line, Grid
```

In [24]:
```
#定义美国单日新增确诊人数折线图的相关设置：
l1 = Line().add_xaxis(# 配置x轴
                      xaxis_data = use_data['date'].values   # 输入x轴数据
                      )


l1.add_yaxis(# 配置y轴
             series_name = "单日新增人数",  # 设置图例名称
             y_axis = use_data['today_confirm'].values.tolist(),  # 输入y轴数据
             symbol_size = 10, # 设置点的大小
             label_opts = opts.LabelOpts(is_show=False), # 标签设置项，显示标签
             linestyle_opts = opts.LineStyleOpts(width=1.5, type_='dotted'), # 线条宽度和样式
             is_smooth = True, # 绘制平滑曲线
             )

# 设置全局配置项
l1.set_global_opts(title_opts = opts.TitleOpts(title = "3月美国单日新增人数与股票指数涨幅对比折线图",
                                               pos_left = "center"), # 设置图标题和位置
                   axispointer_opts = opts.AxisPointerOpts(is_show = True,
                                                           link = [{"xAxisIndex": "all"}]),  # 坐标轴指示器配置
                   # x轴配置项
                   xaxis_opts = opts.AxisOpts(type_ = "category",
                                              boundary_gap = True), # 坐标轴两边是否留白
                   # y轴配置项
                   yaxis_opts = opts.AxisOpts(name = "单日新增人数"), # 轴标题
                   # 图例配置项
                   legend_opts = opts.LegendOpts(pos_left  ='7%') # 图例的位置
                  )
```

Out[24]: <pyecharts.charts.basic_charts.line.Line at 0x1e64eaf4070>

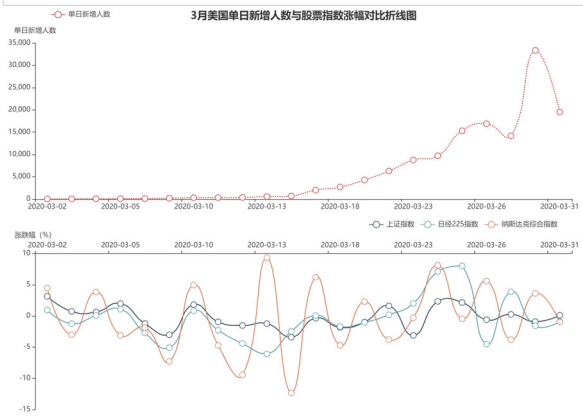
In [25]:
```
#定义三支股票指数变化的折线图设置；
l2 = Line().add_xaxis(xaxis_data = use_data['date'].values)

l2.add_yaxis(series_name = "上证指数",
             y_axis = stock['SSEC'].values,  # 添加上证指数数据
             symbol_size = 10,
             label_opts = opts.LabelOpts(is_show = False),
             linestyle_opts = opts.LineStyleOpts(width = 1.5), # 设置线宽
             is_smooth = True)

l2.add_yaxis(series_name = "日经225指数",
             y_axis = stock['N225'].values,  # 添加日经225指数数据
             symbol_size = 10,
             label_opts = opts.LabelOpts(is_show = False),
             linestyle_opts = opts.LineStyleOpts(width = 1.5),
             is_smooth = True)

l2.add_yaxis(series_name = "纳斯达克综合指数",
             y_axis = stock['NASDAQ'].values, # 添加纳斯达克综合指数数据
             symbol_size = 10,
             label_opts = opts.LabelOpts(is_show = False),
             linestyle_opts = opts.LineStyleOpts(width = 1.5),
             is_smooth = True)

l2.set_global_opts(axispointer_opts = opts.AxisPointerOpts(  # 设置坐标轴指示器
                                                           is_show = True,
                                                           link = [{"xAxisIndex": "all"}]),  # 对x轴所有索引进行联动

                   xaxis_opts = opts.AxisOpts(grid_index = 1,  # x轴开始的索引
                                              type_ = "category", # 类型
                                              boundary_gap = True,
                                              position = "top", # 坐标轴位置
                                              axisline_opts = opts.AxisLineOpts(is_on_zero=True)),  # x轴或y轴的轴线是否在另一个轴的0刻度上

                   yaxis_opts = opts.AxisOpts(is_inverse = False, name = "涨跌幅（%）",name_gap = 25), # 轴线设置
                   legend_opts = opts.LegendOpts(pos_bottom = '50%',pos_right = '70') # 图例设置
                   )
```

Out[25]: <pyecharts.charts.basic_charts.line.Line at 0x1e64eb08430>

将两幅图按照上下位置进行组合：

In [26]:
```
# 绘制组合图形
grid = Grid(init_opts = opts.InitOpts(width = "1024px", height = "768px")) # 设置图形的长和宽

grid.add(chart=l1,  # 添加第一个图表
         grid_opts = opts.GridOpts(pos_left = 50, pos_right = 50, height = "35%"))  # 直角坐标系网格配置项

grid.add(chart = l2, # 添加第二个图表
         grid_opts = opts.GridOpts(pos_left = 50, pos_right = 50, pos_top = "55%", height = "35%"))

# 利用notebook进行渲染
grid.render_notebook()
```

Out[26]:



Type *Markdown* and LaTeX: α^2