数据可视化 HW5

钟诚 16307110259

May 26, 2020

1 设计灰度向彩色（伪彩）变换的算法，实现代码并进行测试

从灰度值向伪彩变换时，涉及到单通道向多通道的映射，如果对应的算法不合适可能会影响到图片本身的美感，所以本次实验中我为了在体现效果的同时保证图片的美感，设置了较为简单的映射关系。

代码实现如下：

```
def Gray2RGB():
    # 从灰度到伪彩的对应关系
    def RGBtrans(pixel):
        if 0 < pixel < 64:
            return [255, 4*pixel, 0]
        elif 64 <= pixel < 128:
            return [510-4*pixel, 255, 0]
        elif 128 <= pixel < 192:
            return [0, 255, 4*pixel-510]
        else:
            return [0, 1020 - 4*pixel, 255]

    # 打开灰度图像
    img = Image.open('Gray_zibin.jpg').convert('L')
    img = np.array(img)
    width, height = np.shape(img)
    new = np.zeros((width, height,3))
    # 通过对应关系计算出RGB三通道对应的值
    for i in range(width):
        for j in range(height):
            new[i, j, :] = RGBtrans(img[i, j])
    plt.imshow(new)
    # 显示图像
    plt.show()
```

实验效果如下：从变换结果中可见，从灰度到伪彩的变换保留了图片本身的轮廓信息，如果想要实现更精确的颜色变换，可以从 RGB 三通道中通过算法尝试实现对灰度值最相近亮度值的映射。

(a) 灰度图像　　(b) 伪彩图像

Figure 1: 从灰度图像到伪彩图像的变化

2 请使用世界各国 GDP 总量数据
(1) 用折线、散点做一个完整可视化图，显示世界各国 20 年的 GDP 数值
(2) 使用地图做图，显示世界各国 GDP 在 20 年来的动态变化

2.1 使用折线，散点显示 GDP 数值

在读取了数据后，我们便可以选取相应的国家进行作图，这里选取了中国，美国，英国，日本，法国五个国家进行作图，从图像中可以看出，美国的 GDP 一直稳居五国之首，而中国的 GDP 从 2006 年开始上升势头明显，从 2010 年起已经稳居五国第二位。折线图和代码如下：

```
def GDPplot():
    # 打开文件
    workbook = xlrd.open_workbook("GDP-fromworldbank.xls")
    worksheet = workbook.sheet_by_index(0)
    nrows = worksheet.nrows
    title = worksheet.row_values(3)
    GDP = []
    # 设定需要绘制折线图的国家和对应的颜色
    chart_country = ['China', 'United States', 'United Kingdom', 'Japan', 'France']
    color = ['red', 'gold','springgreen', 'fuchsia', 'royalblue']
    plot_x = title[-21:-1:]
    fig, ax = plt.subplots()


    xticks=list(range(0,len(plot_x),1))
    xlabels=[plot_x[x] for x in xticks]

    ax.set_xticks(xticks)
    ax.set_xticklabels(xlabels, rotation=40)
    #ax.set_yscale("log")

    for i in range(5, nrows):
        data = worksheet.row_values(i)
        GDP.append(data[0:1:] + data[4:-1:])
        country_name = data[0]
        # 如果在读取文件的时候读取到了对应国家，则进行绘图
```

```
        if country_name in chart_country:
            index = chart_country.index(country_name)
            value = data[-21:-1:]
            plt.plot(plot_x, value, label = country_name, color = color[index])
    plt.legend()
    plt.title(' Line chart of GDP in five countries')
    plt.show()
```

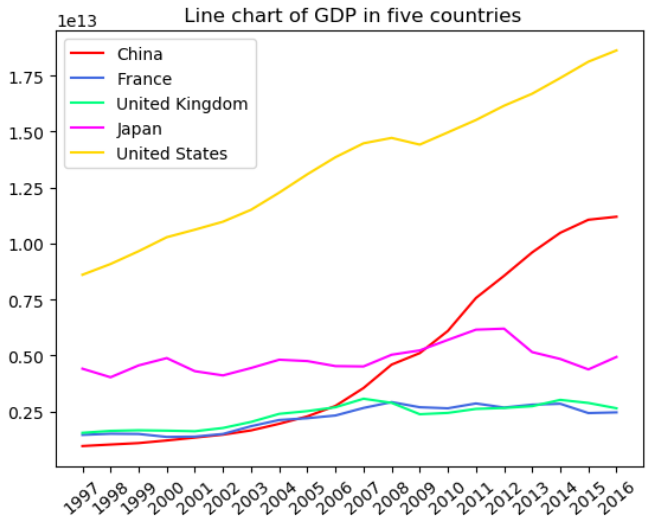


Figure 2: 五国 GDP 折线图

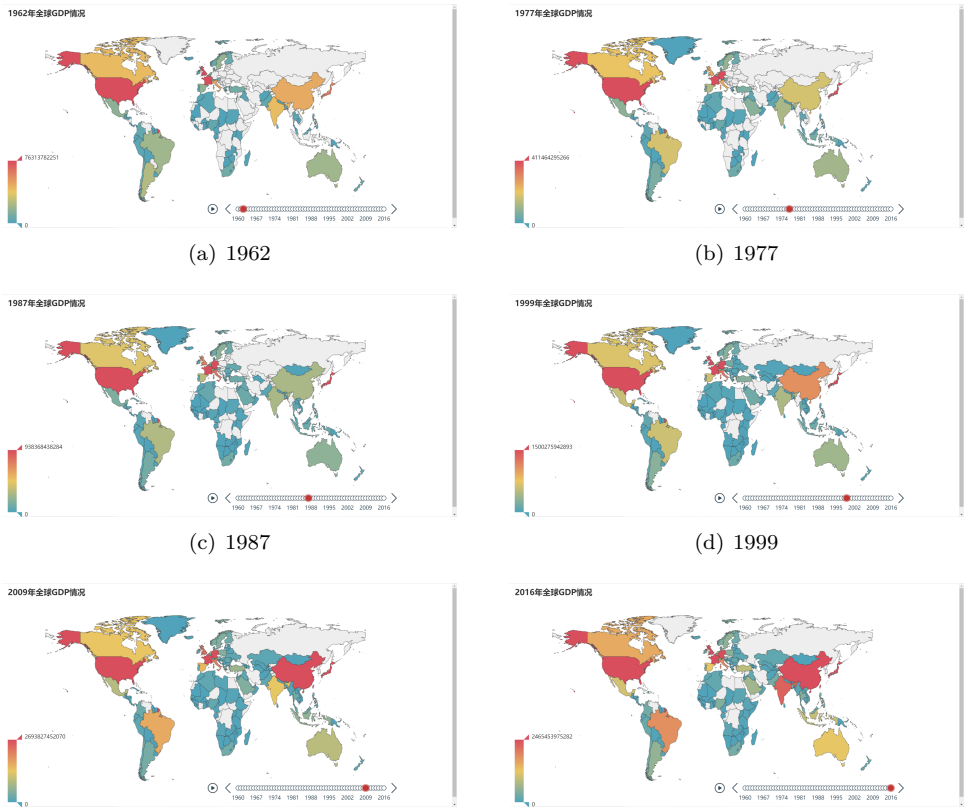
2.2 显示世界各国 GDP 在 20 年来的动态变化

本实验中，为了体现世界各国 GDP20 年内的动态变化，我使用了 **pyecharts** 的库，通过其中的 **Timeline** 方法实现热力图随时间的动态变化

代码实现和动态变化中的一些示例图如下所示，我这里显示了 40 年的 GDP 变化，可以看到中国在 40 年中在全球的 GDP 比重大幅上升，老牌资本主义国家也始终保持在全球的领先地位。更详细的变化图在随报告附带的 html 中。

```
def GDPplot():
    '''......'''
    # GDP timeline
    # 绘制动态变化图
    # 实例化时间轴
    timeline = Timeline()
    timeline.add_schema(pos_left="50%", pos_right="10px", pos_bottom="15px")
    country_names = worksheet.col_values(0)[4::]
    # 去掉数据中"world"的部分以免数值差过大
    del country_names[-7]
```

```
for i in range(len(plot_x)):
    # 将不同的年份做热力图并添加到时间轴中
    GDPCount = worksheet.col_values(i+4)[4::]
    del GDPCount[-7]
    GDPCountint = []
    for j in GDPCount:
        if j == '':
            GDPCountint.append(0)
        else:
            GDPCountint.append(int(j))
    maxGDP = GDPCountint[75]
    years = int(i) + 1960
    maps = Map( init_opts=opts.InitOpts(width="1900px", height="900px",
        bg_color="#ADD8E6", page_title= str(years) + "年全球GDP情况",theme="white"))
    # 添加各个地区的GDP到热力图中
    maps.add("GDP",[list(z) for z in zip(country_names,
        GDPCount)],is_map_symbol_show=False, maptype="world",
        label_opts=opts.LabelOpts(is_show=False),
        itemstyle_opts=opts.ItemStyleOpts(color="rgb(49,60,72)"))
    maps.set_global_opts(title_opts = opts.TitleOpts(title=str(years) +
        "年全球GDP情况"), legend_opts=opts.LegendOpts(is_show=False), visualmap_opts
        = opts.VisualMapOpts(max_=maxGDP))
    timeline.add(maps, str(years))
# 将结果导出到html中
timeline.render("GDP变化图.html")
```



(a) 1962 (b) 1977

(c) 1987 (d) 1999

(e) 2009 (f) 2016

Figure 3: 动态图中部分年份 GDP 热力图示例

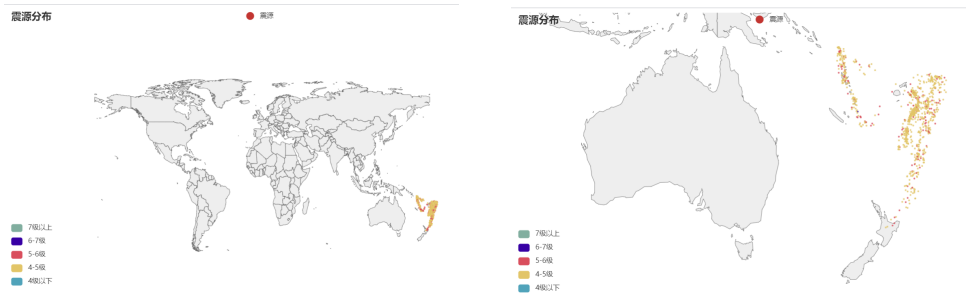
3 请使用地震数据，使用地图可视化的方法对数据进行可视化，展现地震的地点。

与第二题相同，通过经纬度数据，我们可以在世界地图上对地震的地点进行标记，代码实现和代码结果如下，可以看到地震的区域主要分布在新西兰附近，出于大平洋板块和印度洋板块的交界之处，具体结果也在随报告附上的 html 中

```
def quakes():
    # 对地震数据进行绘图
    quakes_data = pd.read_csv("quakes.csv")
    latitude = quakes_data['lat'].values
    longitude = quakes_data['long'].values
    mag = quakes_data['mag'].values
    index = list(range(len(mag)))

    #可视化
    geo = Geo()
    geo.add_schema(maptype='world')
    for i in range(len(index)):
        geo.add_coordinate(index[i], longitude[i], latitude[i])
```

```
data_pair = [list(z) for z in zip(index, mag)]
# 将数据添加进图中
geo.add('震源', data_pair,symbol_size=3)
geo.set_series_opts(label_opts=opts.LabelOpts(is_show=False))
# 设置显示区间
pieces = [
       {'max': 4, 'label': '4级以下', 'color': '#50A3BA'},
       {'min': 4, 'max': 5, 'label': '4-5级', 'color': '#E2C568'},
       {'min': 5, 'max': 6, 'label': '5-6级', 'color': '#D94E5D'},
       {'min': 6, 'max': 7, 'label': '6-7级', 'color': '#3700A4'},
       {'min': 7, 'label': '7级以上', 'color': '#81AE9F'},
   ]
geo.set_global_opts(
       visualmap_opts=opts.VisualMapOpts(is_piecewise=True, pieces=pieces),
       title_opts=opts.TitleOpts(title="震源分布")
   )
#导出图像
geo.render('地震地点分布.html')
```



(a) 世界地图　　(b) 放大后地图

Figure 4: 地震震源分布