20377199 赵芮箐 第7周作业

经济管理中通常有大量的数据以csv等结构化格式存在，如本次作业要用的空气质量数据。数据见在线平台，格式说明如https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data。

1. 至少实现一个数据分析类，以提供数据的读取及基本的时间（如某区域某类型污染物随时间的变化）和空间分析（某时间点或时间段北京空气质量的空间分布态势）方法。

2. 至少实现一个数据可视化类，以提供上述时空分析结果的可视化，如以曲线、饼、地图等形式对结果进行呈现。

3. 如果数据中包含空值等异常值（可人工注入错误数据以测试异常抛出与处理的逻辑），在进行数据分析以及可视化前需要检查数据。因此需要实现NotNumError类，继承ValueError，并加入新属性region, year，month，day,hour，pollutant，对数据进行检测，若取到的一列数据中包含空值等明显错误，则抛出该异常，并提供异常信息。在此基础上，利用try except捕获该异常，打印异常信息，并对应位置的数据进行适当的填充。

4. （附加）污染物含量与气象状态本身是否有相关性？请丰富数据分析类和数据可视化类，增加关于这些相关性探索的方法。

5. （附加）思考不同区域时间变化的趋势及差异的管理意义。

提示：了解并使用csv（标准库），openpyxl等来读取csv等结构化文件，或直接视为一般文本文件读取。

任务准备：数据集介绍

该数据集来自于北京市环境检测中心，包括了2013年3月1日到2017年2月28日，12个国家控制的空气质量监测站每小时的空气污染数据，且每个空气质量站点的气象数据都与中国气象局站相匹配。此空气污染数据集包括12个站点数据集，每个数据集包括了35064个时间数据，共有（35064*12）个样本。

变量包括：PM2.5、PM10、SO2（二氧化硫）、NO2（二氧化氮）、CO（一氧化碳）、O3（臭氧）为空气污染物；剩余变量TEMP（温度）、PRES（大气压）、DEWP（露点温度）、RAIN（降雨量）、WD（风向）、WSPM（风速）天气情况。

站点包括：万寿西宫（西城区）、官园（西城区）、万柳（海淀区）、天坛（东城区）、农展馆（朝阳区）、奥体中心（朝阳区）、怀柔、古城（顺义区）、顺义、东寺（平谷区）、定陵（昌平区）、昌平共十二个，其地理位置分布如下：

> 是的，是我剽窃来的，嘿嘿，在我搜WSPM是啥的时候发现有人用了同样的数据集，说的挺好的就搬过来啦~
>
> PS：但感觉 "Dongsi" 应该是【东四街道】而非【东寺】

任务一：实现数据分析类

```
class Data_analyze:
    '''
    数据分析类
    '''
    def __init__(self, dir_path):
        """
        数据的读取

        :param dir_path:数据文件所在文件夹
        """
        self._df = []                                                    # 每个csv文件按df格式储存进列表
        self._region = ["Aotizhongxin","Changping","Dingling","Dongsi","Guanyuan","Gucheng","Huairou","Nongzhanguan","Shunyi","Tiantan","Wanliu","Wanshouxigong"]
        self._pollutant = ["PM2.5","PM10","SO2","NO2","CO","O3"]
```

```
        self._climate = ["TEMP","PRES","DEWP","RAIN","wd","WSPM"]

        print("-----数据读取中，请稍后-----")
        files = os.listdir(dir_path)                                    # 获取目录下所有的文
件
        for path in files:
            with open(dir_path + "/" + path) as f:
                df = pd.read_csv(f)
                df = df.set_index("No")
                self._df.append(df)
        print("-----数据读取完成-----")

    def time_analyze(self, pol, region):
        """
        时间分析：某区域某类型污染物随时间的变化(PS:以均值代表每天的污染物含量)

        :param pol:污染物类型
        :param region:某区域
        :return pol_time:某污染物在该地区每一天的含量值
        """
        data_time = []
        df = self._df[self._region.index(region)]
        for n in range(1,len(df)+1,24):
            time = date(df.at[n,'year'], df.at[n,'month'], df.at[n,'day'])  # 获
得该日期
            pol_mean = df[pol][n-1:n+23].mean()                             # 获
得该天下污染物的均值
            data_time.append([time, pol_mean])
        pol_time = pd.DataFrame(data_time, columns=['Date', pol])           # 将
时间存为时间序列索引，方便后续画图
        pol_time = pd.DataFrame(pol_time).set_index('Date')
        pol_time.index = pd.to_datetime(pol_time.index)
        return pol_time

    def space_analyze(self, pol, *time):
        """
        空间分析：某时间点北京空气质量的空间分布态势(PS:以均值代表某时间点的污染物含量)

        :param pol:污染物类型
        :param *time:某时间点，不定长列表，可以表示年月日
        :return pol_space:某污染物在某时间点各区域的含量值
        """
        data_space = []
        if len(time) == 1:
            for df in self._df:
                region = df.iloc[0,-1]
                df_year = df[df['year'] == time[0]]
                pol_mean = df_year[pol].mean()
                data_space.append([region, pol_mean])
        elif len(time) == 2:
            for df in self._df:
                region = df.iloc[0,-1]
                df_month = df[(df['year'] == time[0]) & (df['month'] ==
time[1])]
                pol_mean = df_month[pol].mean()
```

```
                data_space.append([region, pol_mean])
        else:
            for df in self._df:
                region = df.iloc[0,-1]
                df_day = df[(df['year'] == time[0]) & (df['month'] == time[1]) &
(df['day'] == time[2])]
                pol_mean = df_day[pol].mean()
                data_space.append([region, pol_mean])
        pol_space = pd.DataFrame(data_space, columns=['Region', pol])
        pol_space = pd.DataFrame(pol_space).set_index('Region')
        return pol_space

    def cor_analyze(self):
        df_all = pd.concat(self._df)             # 拼接所有数据表
        df_pol_cli = df_all.iloc[:,4:-1]
        corr = df_pol_cli.corr().iloc[:-5,6:]    # 得到污染物和气候的相关系数矩阵
        return corr
```

结果展示：

- 时间统计：

  > 以 Aotizhongxin地区，PM2.5数据为例

Date	PM2.5
2013-03-01	7.125000
2013-03-02	30.750000
2013-03-03	76.916667
2013-03-04	22.708333
2013-03-05	148.875000
...	...
2017-02-24	21.541667
2017-02-25	11.208333
2017-02-26	28.125000
2017-02-27	71.954545
2017-02-28	13.166667

- 空间统计：

  > 以 2015年12月，SO2数据为例

Region	SO2
Aotizhongxin	23.637108
Changping	14.098361
Dingling	15.607192
Dongsi	23.708277
Guanyuan	22.748982
Gucheng	23.259511
Huairou	9.987738
Nongzhanguan	23.207880
Shunyi	14.129760
Tiantan	12.735135
Wanliu	25.440977
Wanshouxigong	24.500000

- 相关关系：

	TEMP	PRES	DEWP	RAIN	WSPM
PM2.5	-0.131127	0.018566	0.114656	-0.014359	-0.272205
PM10	-0.096209	-0.017971	0.070310	-0.026519	-0.183665
SO2	-0.321799	0.223236	-0.266781	-0.040241	-0.108717
NO2	-0.278192	0.174167	-0.031599	-0.043785	-0.400460
CO	-0.326237	0.188195	-0.057129	-0.013342	-0.297511
O3	0.594910	-0.445961	0.312074	0.023320	0.295743

任务二：实现数据可视化类

```
class Data_view(Data_analyze):
    '''
    数据可视化类
    '''
    def __init__(self, dir_path):
        super().__init__(dir_path)

    def time_view(self, pol, region):
        """
        时间分析可视化：画折线图以及热力图
        """
        #画折线图
        df = super().time_analyze(pol, region)
        df_month = df.resample("M").mean()

        plt.subplot(2, 1, 1)
        df[pol].plot()
        plt.xlabel("")
        plt.ylabel(f'{pol} day average')
        plt.title(f'{pol} in {region}')

        plt.subplot(2, 1, 2)
        df_month[pol].plot()
        plt.xlabel("")
        plt.ylabel(f'{pol} month average')
        plt.show()

        #画热力图
        df["year"] = pd.DatetimeIndex(df.index).year
        df["month"] = pd.DatetimeIndex(df.index).month
        df_cal = df.pivot_table(index="month", columns="year", values=pol,
aggfunc=np.mean)
        ax = sns.heatmap(df_cal, cmap='RdYlGn_r', robust=True, fmt='.2f',
                 annot=True, linewidths=.5, annot_kws={'size':11},
                 cbar_kws={'shrink':.8, 'label':pol})
        ax.set_yticklabels(ax.get_yticklabels(), rotation=0, fontsize=10)
        ax.set_xticklabels(ax.get_xticklabels(), rotation=0, fontsize=10)
        plt.title(f'Average {pol} in {region}', fontdict={'fontsize':18},
pad=14)
        plt.show()

    def space_view(self, pol, *time):
        """
```

```
        空间分析可视化：画饼图以及热力图
        """
        #画饼图
        df = super().space_analyze(pol, *time)
        data_list = np.array(df[pol])
        cmap = plt.get_cmap("tab20c")
        color=cmap(range(12))
        plt.pie(data_list, colors = color,
                labels=self._region,
                textprops = {'fontsize':7, 'color':'k'},
                autopct='%.2f%%')
        plt.title(f"Beijing {pol} distribution in {str(time)[1:][:-1]}")
        plt.show()

        #画热力图
        g = Geo(init_opts=opts.InitOpts(width='1000px',
                                height='600px',
                                theme=ThemeType.DARK),)
        g.add_schema(maptype='北京')
        region = ['奥体中心', '昌平', '定陵', '东四', '官园', '古城', '怀柔', '农展馆',
'顺义', '天坛', '万柳', '万寿西宫']
        region_loc = [[39.985069,116.401665],
                      [40.22077,116.23128],
                      [40.286598,116.238896],
                      [39.924995,116.417679],
                      [39.932392,116.355858],
                      [39.911766,116.193359],
                      [40.316,116.63177],
                      [39.939819,116.46846],
                      [40.13012,116.65477],
                      [40.029076,116.311478],
                      [39.967056,116.296959],
                      [39.879616,116.36853]]
        for i in range(len(region)):
            # 定义坐标对应的名称，添加到坐标库中 add_coordinate(name, lng, lat)
            g.add_coordinate(region[i], region_loc[i][1], region_loc[i][0])
        # 将数据添加到地图上
        g.add(  series_name = pol,                                  # 系列名称
                data_pair = list(zip(region, data_list)),       # 数据项 (坐标点名
称，坐标点值)
                blur_size = 20,
                symbol_size = 15,
                type_ = ChartType.HEATMAP                           #类型选为热力图
                #type_ = ChartType.EFFECT_SCATTER
            )
        # 设置样式
        g.set_series_opts(label_opts=opts.LabelOpts(is_show=False))
        # 设置标题及分段
        g.set_global_opts(
                      visualmap_opts=opts.VisualMapOpts(max_= max(data_list),
is_piecewise=True),
                      title_opts=opts.TitleOpts(title="北京空气质量分布"))
        # 渲染
        g.render(f"week7/Beijing {pol} distribution in {str(time)[1:]
[:-1]}.html")
```

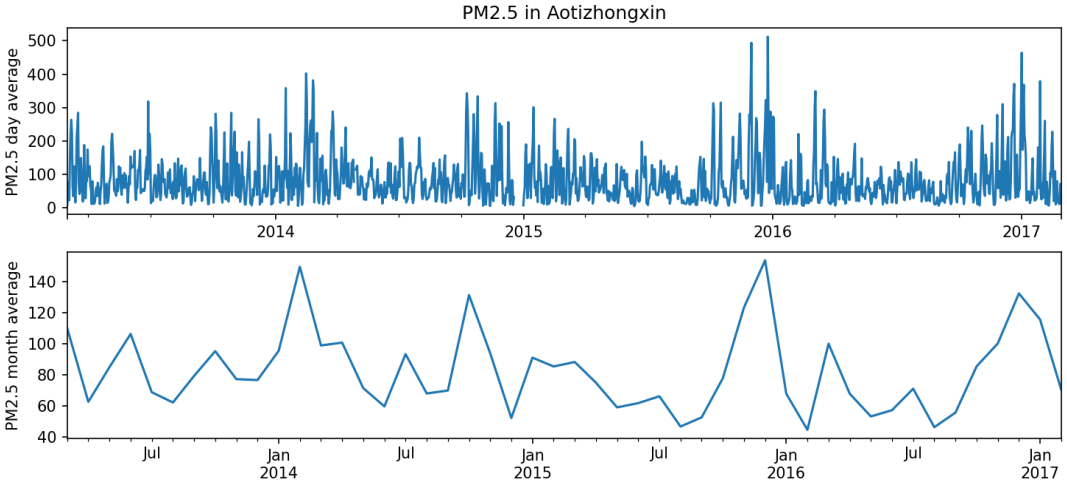
```
    def cor_view(self):
        corr = super().cor_analyze()
        heatmap = sns.heatmap(corr, cmap="RdBu_r")
        plt.title('Covariance Matrix', fontdict={'fontsize':18})
        plt.show()
```

结果展示：

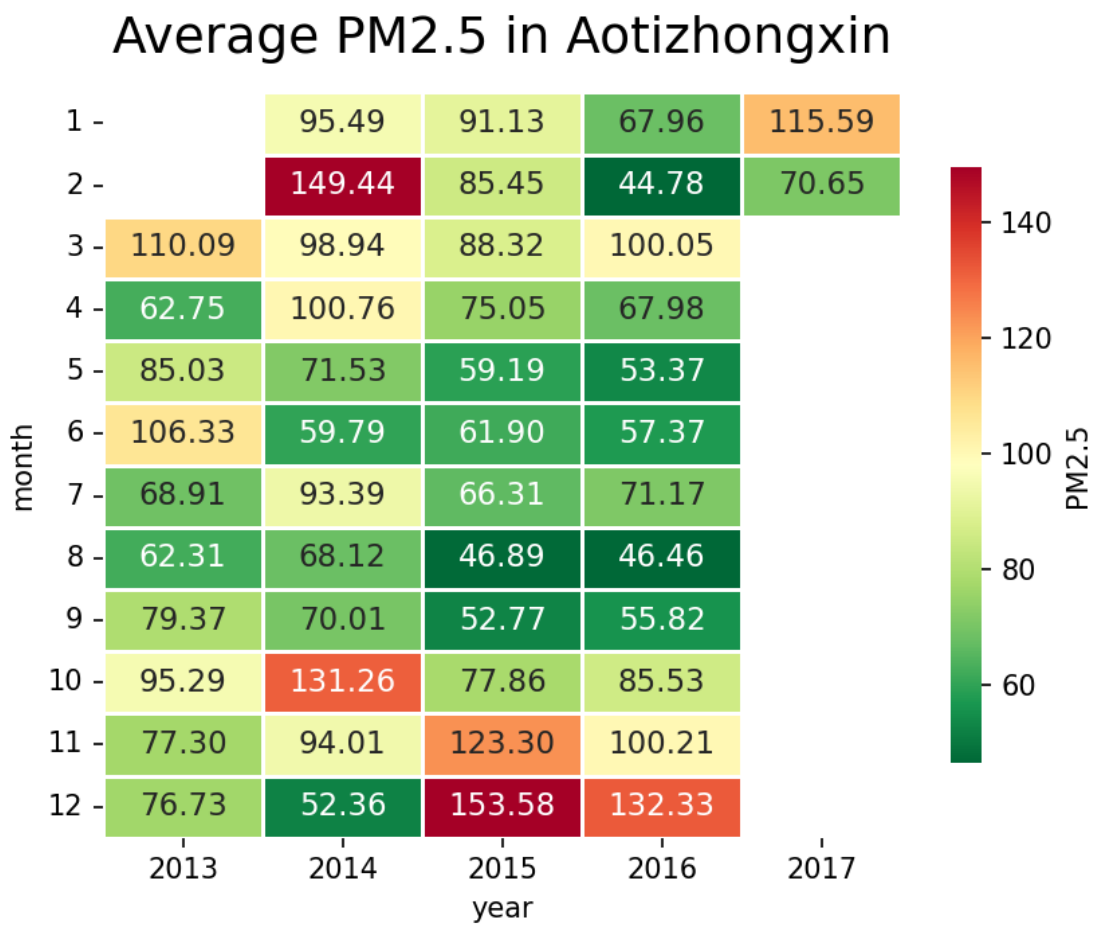
- 时间统计：

  > 以 Aotizhongxin地区，PM2.5数据为例

  - 按日和月分别画了个变化趋势图



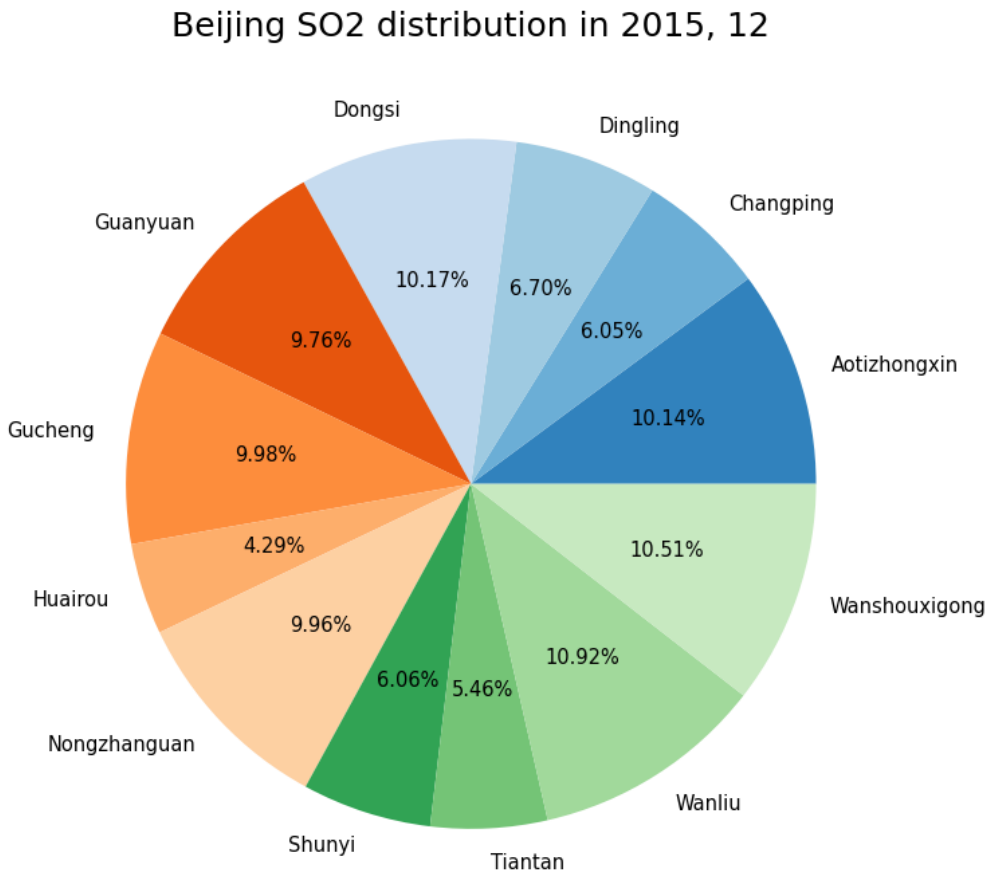
  - 热力图



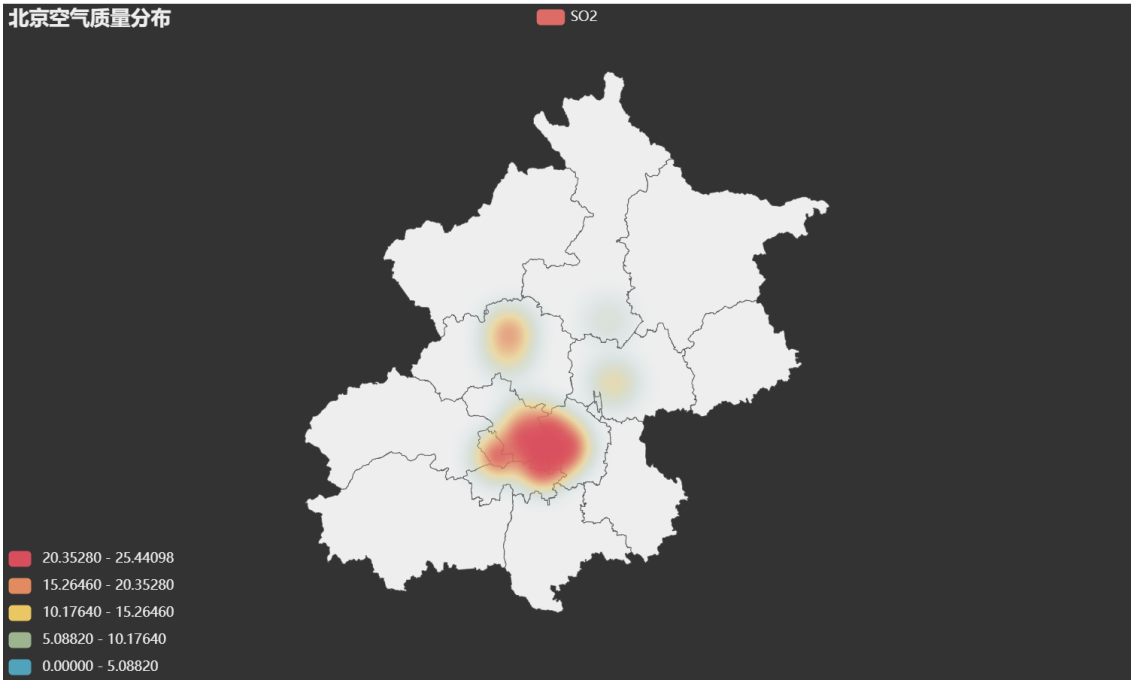
- 空间统计：

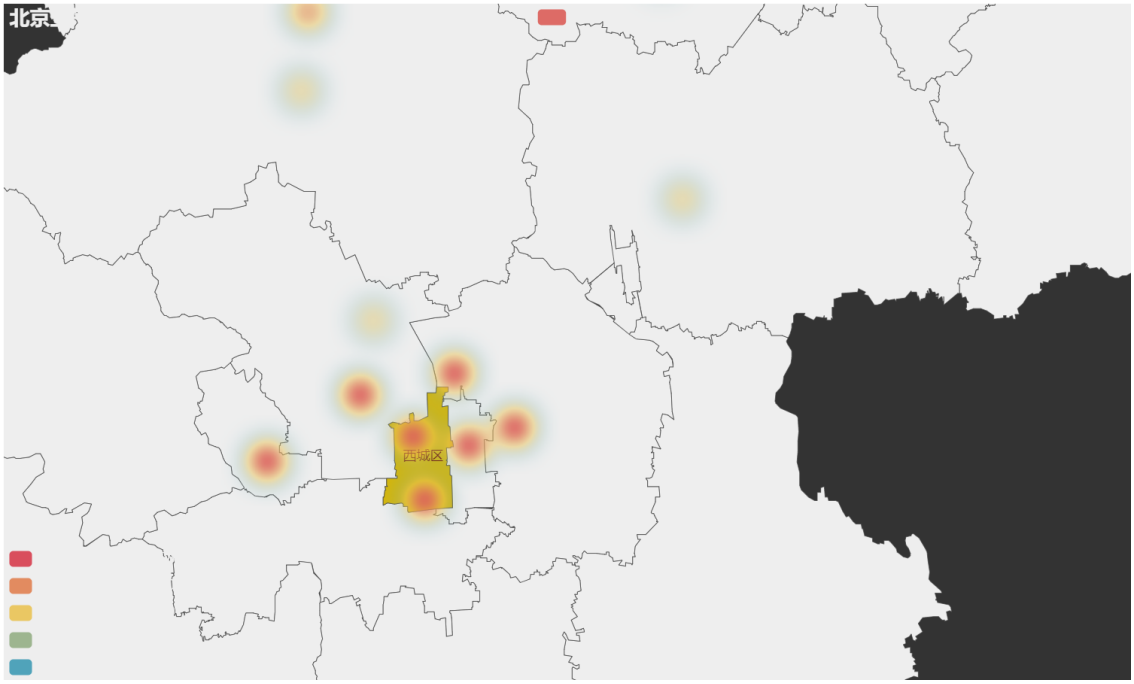
以 Aotizhongxin地区，PM2.5数据为例

- 饼图



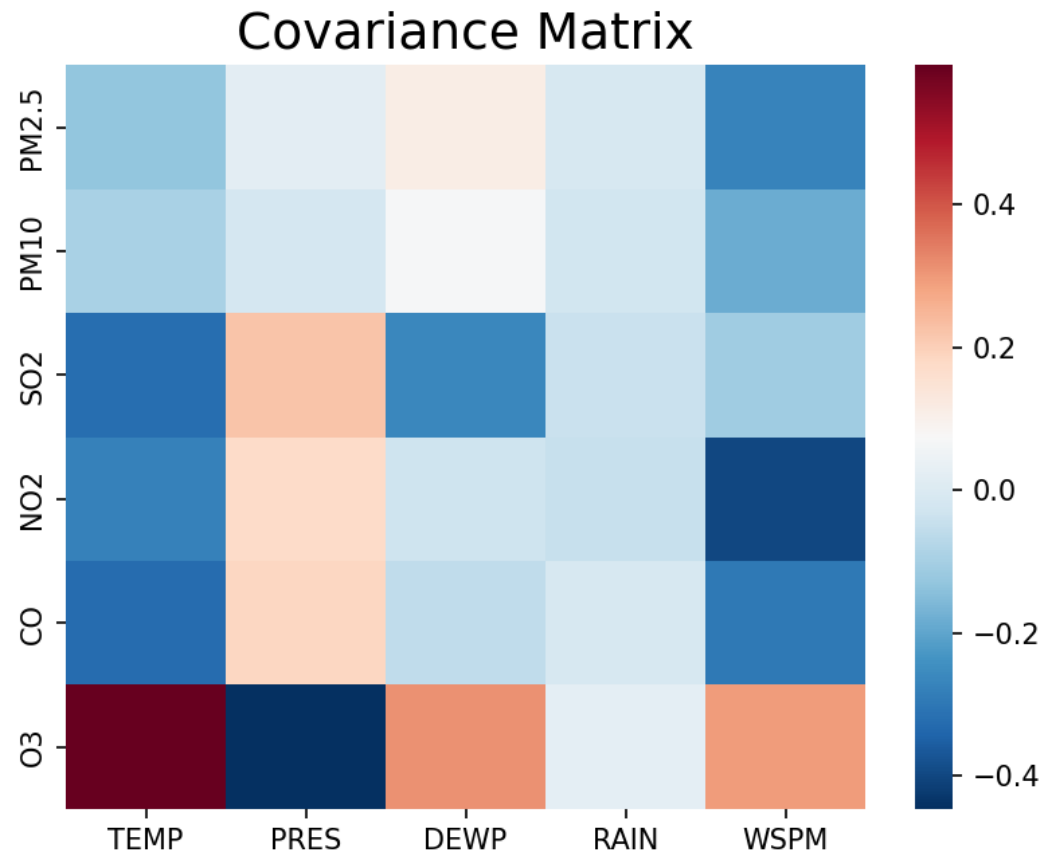
- 热力图





> 主城区都...明显严重很多

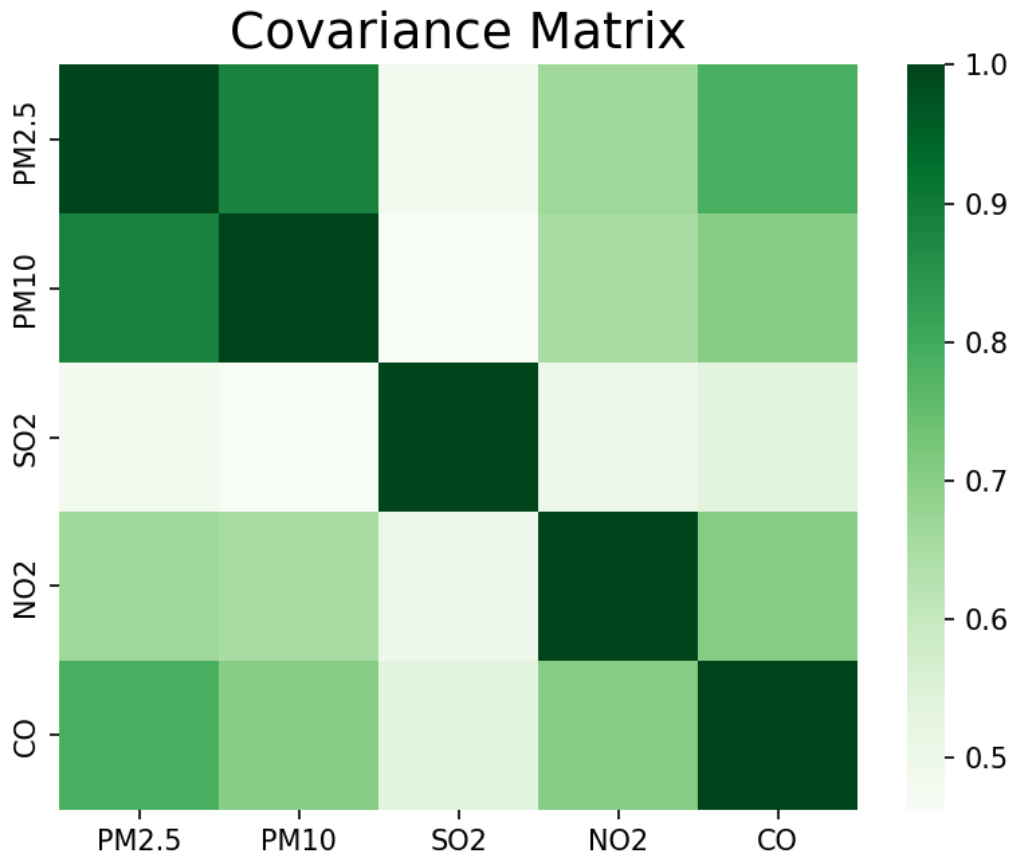
- 相关关系：



比较显著的是：臭氧和温度呈明显正相关和大气压成负相关

> 科学上：高温天气下，温度越高，越容易产生臭氧，同时温度越高，大气压越低。

感觉相比于给出的污染物与气候之间的关系，污染物本身的相关性其实挺大的：



任务三：异常值处理

```
class NotNumError(ValueError):
    '''
    空值异常值类
    '''
    def __init__(self, region, year, month, day, hour, pollutant):
        self._region = region
        self._year = year
        self._month = month
        self._day = day
        self._hour = hour
        self._pollutant = pollutant
        self._message = f"{region} Sheet: In {year}-{month}-{day} {hour}h the
{pollutant} data has NotNumError."

class NotNumTest(Data_analyze):
    '''
    空值异常测试类
    '''
    def __init__(self, dir_path):
        super().__init__(dir_path)

    def examine(self):
        print("-----正在检查数据，请稍后-----")
        for df in self._df:
            #df = df[:100]
            for No in range(1, len(df)+1):
                row = np.array(df.loc[No]).tolist()
                try:
```

```
                    region = row[-1]; year = row[0]; month = row[1]; day =
row[2]; hour = row[3]
                    pol_data = row[5:10]
                    for pol in pol_data:
                        if np.isnan(pol) == True:
                            pollutant = self._pollutant[pol_data.index(pol)]
                            # 用该缺失值这一天的均值代替
                            df_day = df[(df['year'] == year) & (df['month'] ==
month) & (df['day'] == day)]
                            df.at[No, pollutant] = df_day[pollutant].mean()
                            raise NotNumError(region, year, month, day, hour,
pollutant)
                except NotNumError as nne:
                    print(nne._message)
            print(f"{region} Sheet finishes examining.")
```

结果展示：

```
Aotizhongxin Sheet: In 2013-3-4 2h the PM10 data has NotNumError.
Aotizhongxin Sheet: In 2013-3-4 3h the NO2 data has NotNumError.
Aotizhongxin Sheet: In 2013-3-4 4h the SO2 data has NotNumError.
Aotizhongxin Sheet finishes examining.
Changping Sheet: In 2013-3-2 2h the PM10 data has NotNumError.
Changping Sheet: In 2013-3-2 3h the NO2 data has NotNumError.
Changping Sheet: In 2013-3-2 4h the SO2 data has NotNumError.
Changping Sheet finishes examining.
Dingling Sheet: In 2013-3-1 0h the SO2 data has NotNumError.
Dingling Sheet: In 2013-3-1 1h the SO2 data has NotNumError.
Dingling Sheet: In 2013-3-1 3h the SO2 data has NotNumError.
Dingling Sheet: In 2013-3-1 4h the SO2 data has NotNumError.
Dingling Sheet: In 2013-3-1 10h the PM2.5 data has NotNumError.
Dingling Sheet: In 2013-3-1 11h the SO2 data has NotNumError.
Dingling Sheet: In 2013-3-1 12h the PM2.5 data has NotNumError.
Dingling Sheet: In 2013-3-1 14h the SO2 data has NotNumError.
Dingling Sheet: In 2013-3-1 15h the SO2 data has NotNumError.
Dingling Sheet: In 2013-3-1 16h the SO2 data has NotNumError.
Dingling Sheet: In 2013-3-1 17h the SO2 data has NotNumError.
Dingling Sheet: In 2013-3-1 18h the SO2 data has NotNumError.
Dingling Sheet: In 2013-3-1 19h the SO2 data has NotNumError.
Dingling Sheet: In 2013-3-1 20h the SO2 data has NotNumError.
Dingling Sheet: In 2013-3-1 21h the SO2 data has NotNumError.
Dingling Sheet: In 2013-3-1 22h the SO2 data has NotNumError.
```

> PS：为方便结果展示，每个表就取了前一百行数据

思考：不同区域时间变化的趋势及差异的管理意义

- 更了解污染物及气候的季节性变化，对未来的值进行更好的预测
- 有针对性的对工厂等实施污染物排放的管控政策

代码：

https://github.com/rachhhhing/mp2022_python/blob/master/week7/week7.py

Ref:

- csv标准库：https://docs.python.org/zh-cn/3/library/csv.html

  openpyxl：https://www.cnblogs.com/hls-code/p/15674197.html

  > 感觉 pd.read_csv 更好使一些嘿嘿

- datatime：https://www.cnblogs.com/awakenedy/articles/9182036.html
- 时序数据可视化：https://blog.csdn.net/deephub/article/details/109839988
- pyecharts热力图：https://blog.csdn.net/qq_39451578/article/details/104372597

  https://zhuanlan.zhihu.com/p/370741946
- 相关性分析：https://blog.csdn.net/BF02jgtRS00XKtCx/article/details/107218146
- 跟我们用同一份数据，但进行的是回归预测：

  https://blog.csdn.net/weixin_45529837/article/details/108805415