71086032-曾诗仪-第十五周作业

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。数据见在线平台的demo/python15，格式说明如https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data。请利用numpy或pandas等相关库，完成如下任务。

1. 实现一个数据分析类，基于pandas, 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。

```
import pandas as pd
import os

class DataAnalysis:
    def __init__(self, data_folder):
        self.data_files = self.get_csv_files(data_folder)  # 获取所有CSV文件路径
        self.data = self.read_csv_files(self.data_files)  # 读取所有CSV文件并合并数据

    def get_csv_files(self, data_folder):
        csv_files = []
        for file in os.listdir(data_folder):
            if file.endswith(".csv"):
                csv_files.append(os.path.join(data_folder, file))
        return csv_files

    def read_csv_files(self, csv_files):
        dfs = []
        for file in csv_files:
            df = pd.read_csv(file)
            dfs.append(df)
        if dfs:
            return pd.concat(dfs, ignore_index=True)  # 合并所有数据
        else:
            return None

    def pollutant_over_time(self, region, pollutant_type):
        region_data = self.data[self.data['station'] == region]  # 筛选指定区域的数据
        pollutant_data = region_data[['year', 'month', 'day', 'hour', pollutant_type]]  # 获取指定污染物类型的时间和数值数据
        pollutant_data.loc[:, 'time'] = pd.to_datetime(pollutant_data[['year', 'month', 'day', 'hour']]).values# 创建时间列
        pollutant_data.set_index('time', inplace=True)  # 将时间列设置为索引
        return pollutant_data

    def air_quality_distribution(self, time_point):
        time_data = self.data[(self.data['year'] == time_point.year) &
                              (self.data['month'] == time_point.month) &
                              (self.data['day'] == time_point.day) &
```

```
                                (self.data['hour'] == time_point.hour)]  # 筛选
指定时间点的数据
        return time_data[['station', 'PM2.5', 'PM10', 'SO2', 'NO2', 'CO',
'O3', 'TEMP', 'PRES', 'DEWP', 'RAIN', 'wd', 'WSPM']]

# 示例用法
data_folder = 'C:\\Users\\shiye\\Desktop\\data'
analysis = DataAnalysis(data_folder)

# 时间分析示例
region = 'Aotizhongxin'
pollutant_type = 'PM2.5'
pollutant_data = analysis.pollutant_over_time(region, pollutant_type)
print(pollutant_data)

# 空间分析示例
time_point = pd.to_datetime('2014-05-01 00:00:00')
air_quality_distribution = analysis.air_quality_distribution(time_point)
print(air_quality_distribution)
```

2. 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。

```
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns


class DataAnalysis:
    def __init__(self, data_folder):
        self.data_files = self.get_csv_files(data_folder)  # 获取所有CSV文件路
径
        self.data = self.read_csv_files(self.data_files)  # 读取所有CSV文件并合
并数据

    def get_csv_files(self, data_folder):
        csv_files = []
        for file in os.listdir(data_folder):
            if file.endswith(".csv"):
                csv_files.append(os.path.join(data_folder, file))
        return csv_files

    def read_csv_files(self, csv_files):
        dfs = []
        for file in csv_files:
            df = pd.read_csv(file)
            dfs.append(df)
        if dfs:
            return pd.concat(dfs, ignore_index=True)  # 合并所有数据
        else:
            return None
```

```
    def pollutant_over_time(self, region, pollutant_type):
        region_data = self.data[self.data['station'] == region]  # 筛选指定区域的数据
        pollutant_data = region_data[['year', 'month', 'day', 'hour', pollutant_type]]  # 获取指定污染物类型的时间和数值数据
        pollutant_data.loc[:, 'time'] = pd.to_datetime(pollutant_data[['year', 'month', 'day', 'hour']]).values  # 创建时间列
        pollutant_data.set_index('time', inplace=True)  # 将时间列设置为索引
        return pollutant_data

    def air_quality_distribution(self, time_point):
        time_data = self.data[(self.data['year'] == time_point.year) &
                              (self.data['month'] == time_point.month) &
                              (self.data['day'] == time_point.day) &
                              (self.data['hour'] == time_point.hour)]  # 筛选指定时间点的数据
        return time_data[['station', 'PM2.5', 'PM10', 'SO2', 'NO2', 'CO', 'O3', 'TEMP', 'PRES', 'DEWP', 'RAIN', 'wd', 'WSPM']]


class DataVisualization:
    def __init__(self, analysis):
        self.analysis = analysis

    def plot_pollutant_over_time(self, region, pollutant_type):
        pollutant_data = self.analysis.pollutant_over_time(region, pollutant_type)
        plt.figure(figsize=(10, 6))
        sns.lineplot(data=pollutant_data, x=pollutant_data.index, y=pollutant_type)
        plt.xlabel('Time')
        plt.ylabel(pollutant_type)
        plt.title(f'{pollutant_type} Variation Over Time in {region}')
        plt.show()

    def plot_air_quality_distribution(self, time_point):
        air_quality_distribution = self.analysis.air_quality_distribution(time_point)
        pollutants = ['PM2.5', 'PM10', 'SO2', 'NO2', 'CO', 'O3']
        plt.figure(figsize=(8, 6))
        plt.pie(air_quality_distribution[pollutants].mean(), labels=pollutants, autopct='%1.1f%%')
        plt.title('Air Quality Distribution')
        plt.show()


# 示例用法
data_folder = 'C:\\Users\\shiye\\Desktop\\data'
analysis = DataAnalysis(data_folder)
visualization = DataVisualization(analysis)

# 时间分析示例
region = 'Aotizhongxin'
```

```
pollutant_type = 'PM2.5'
visualization.plot_pollutant_over_time(region, pollutant_type)

# 空间分析示例
time_point = pd.to_datetime('2014-05-01 00:00:00')
visualization.plot_air_quality_distribution(time_point)
```

3. 如果数据中包含空值等异常值，在进行数据分析以及可视化前需要检查数据。可否利用apply等DataFrame相关方法，进行异常值的处理。

   - `pollutant_over_time()`和`air_quality_distribution()`方法进行了修改，使用了`apply`方法来处理缺失值。对于每个需要处理缺失值的列，将`pd.to_numeric()`函数和`fillna()`方法应用到列上，以处理空值和非数值的字符串值。这样可以确保数据在进行数据分析和可视化前得到适当的清洗和处理。

```
        # 处理异常值（空值和无法转换为数字的字符串值）
        pollutants = ['PM2.5', 'PM10', 'SO2', 'NO2', 'CO', 'O3']
        for pollutant in pollutants:
            time_data[pollutant] = time_data[pollutant].apply(pd.to_numeric,
errors='coerce')
            time_data[pollutant].fillna(time_data[pollutant].mean(),
inplace=True)
```

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。

```
    def plot_pollution_weather_correlation(self):
        weather_variables = ['TEMP', 'PRES', 'DEWP', 'RAIN', 'wd', 'WSPM']
        correlation_data = self.analysis.correlation_analysis('PM2.5',
weather_variables)

        plt.figure(figsize=(8, 6))
        sns.heatmap(correlation_data, annot=True, cmap='coolwarm', vmin=-1,
vmax=1)
        plt.title('Correlation between PM2.5 and Weather Variables')
        plt.xlabel('Weather Variables')
        plt.ylabel('PM2.5')
        plt.show()
```

完整代码：

```
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

```
class DataAnalysis:
    def __init__(self, data_folder):
        self.data_files = self.get_csv_files(data_folder)  # 获取所有CSV文件路径
        self.data = self.read_csv_files(self.data_files)  # 读取所有CSV文件并合并数
据

    def get_csv_files(self, data_folder):
        csv_files = []
        for file in os.listdir(data_folder):
            if file.endswith(".csv"):
                csv_files.append(os.path.join(data_folder, file))
        return csv_files

    def read_csv_files(self, csv_files):
        dfs = []
        for file in csv_files:
            df = pd.read_csv(file)
            dfs.append(df)
        if dfs:
            return pd.concat(dfs, ignore_index=True)  # 合并所有数据
        else:
            return None

    def pollutant_over_time(self, region, pollutant_type):
        region_data = self.data[self.data['station'] == region].copy()  # 筛选指定
区域的数据并创建副本
        pollutant_data = region_data[['year', 'month', 'day', 'hour',
pollutant_type]].copy()  # 获取指定污染物类型的时间和数值数据的副本

        # 处理异常值（空值和无法转换为数字的字符串值）
        pollutant_data[pollutant_type] =
pollutant_data[pollutant_type].apply(pd.to_numeric, errors='coerce')

 pollutant_data[pollutant_type].fillna(pollutant_data[pollutant_type].mean(),
inplace=True)

        pollutant_data.loc[:, 'time'] = pd.to_datetime(pollutant_data[['year',
'month', 'day', 'hour']]).values  # 创建时间列
        pollutant_data.set_index('time', inplace=True)  # 将时间列设置为索引
        return pollutant_data

    def air_quality_distribution(self, time_point):
        time_data = self.data[(self.data['year'] == time_point.year) &
                              (self.data['month'] == time_point.month) &
                              (self.data['day'] == time_point.day) &
                              (self.data['hour'] == time_point.hour)].copy()  #
筛选指定时间点的数据并创建副本

        # 处理异常值（空值和无法转换为数字的字符串值）
        pollutants = ['PM2.5', 'PM10', 'SO2', 'NO2', 'CO', 'O3']
        for pollutant in pollutants:
            time_data[pollutant] = time_data[pollutant].apply(pd.to_numeric,
errors='coerce')
```

```
            time_data[pollutant].fillna(time_data[pollutant].mean(),
inplace=True)

        return time_data[['station', 'PM2.5', 'PM10', 'SO2', 'NO2', 'CO', 'O3']]

    def correlation_analysis(self, pollutant_type, weather_variables):
        correlation_data = self.data[[pollutant_type] +
weather_variables].copy()

        # 处理异常值（空值和无法转换为数字的字符串值）
        correlation_data[pollutant_type] =
pd.to_numeric(correlation_data[pollutant_type], errors='coerce')
        for weather_variable in weather_variables:
            correlation_data[weather_variable] =
pd.to_numeric(correlation_data[weather_variable], errors='coerce')

        correlation_data.fillna(correlation_data.mean(), inplace=True)

        return correlation_data.corr()

    def pollution_weather_correlation(self):
        pollutants = ['PM2.5', 'PM10', 'SO2', 'NO2', 'CO', 'O3']
        weather_variables = ['temperature', 'pressure', 'humidity',
'wind_speed']

        correlations = {}
        for pollutant in pollutants:
            pollutant_correlations = []
            for weather_variable in weather_variables:
                correlation = self.correlation_analysis(pollutant,
weather_variable).loc[pollutant, weather_variable]
                pollutant_correlations.append(correlation)
            correlations[pollutant] = pollutant_correlations

        return pd.DataFrame(correlations, index=weather_variables)


class DataVisualization:
    def __init__(self, analysis):
        self.analysis = analysis

    def plot_pollutant_over_time(self, region, pollutant_type):
        pollutant_data = self.analysis.pollutant_over_time(region,
pollutant_type)

        plt.plot(pollutant_data.index, pollutant_data[pollutant_type])
        plt.xlabel('Time')
        plt.ylabel(pollutant_type)
        plt.title(f'{pollutant_type} Variation over Time in {region}')
        plt.show()

    def plot_air_quality_distribution(self, time_point):
        air_quality_distribution =
self.analysis.air_quality_distribution(time_point)
```

```
        pollutants = ['PM2.5', 'PM10', 'SO2', 'NO2', 'CO', 'O3']
        plt.pie(air_quality_distribution[pollutants].mean(), labels=pollutants,
autopct='%1.1f%%')
        plt.title(f'Air Quality Distribution at {time_point}')
        plt.show()

    def plot_pollution_weather_correlation(self):
        weather_variables = ['TEMP', 'PRES', 'DEWP', 'RAIN', 'wd', 'WSPM']
        correlation_data = self.analysis.correlation_analysis('PM2.5',
weather_variables)

        plt.figure(figsize=(8, 6))
        sns.heatmap(correlation_data, annot=True, cmap='coolwarm', vmin=-1,
vmax=1)
        plt.title('Correlation between PM2.5 and Weather Variables')
        plt.xlabel('Weather Variables')
        plt.ylabel('PM2.5')
        plt.show()

# 示例用法
data_folder = 'C:\\Users\\shiye\\Desktop\\data'
analysis = DataAnalysis(data_folder)
visualization = DataVisualization(analysis)

# 时间分析示例
region = 'Aotizhongxin'
pollutant_type = 'PM2.5'
visualization.plot_pollutant_over_time(region, pollutant_type)

# 空间分析示例
time_point = pd.to_datetime('2014-05-01 00:00:00')
visualization.plot_air_quality_distribution(time_point)

# 相关性分析示例
visualization.plot_pollution_weather_correlation()
```

输出结果：

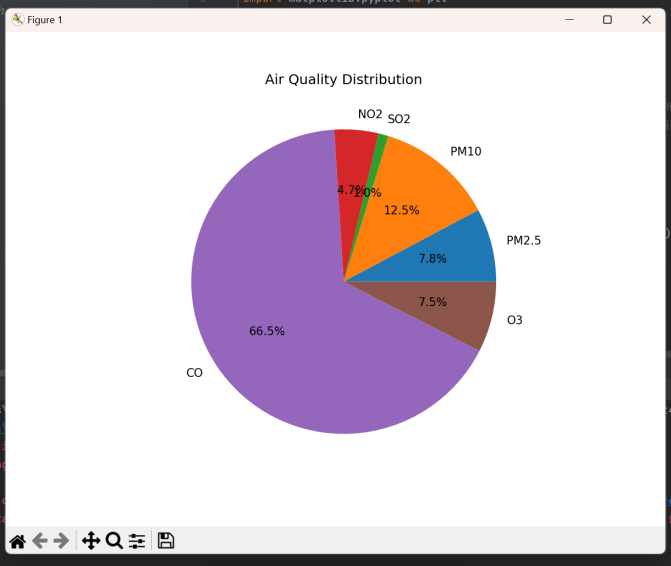
```
            time_data = self.data[(self.data['year'] == time_point.year) &
                                  (self.data['month'] == time_point.month) &
                                  (self.data['day'] == time_point.day) &
                                  (self.data['hour'] == time_point.hour)]  # 筛选指定时间点的数据
            return time_data[['station', 'PM2.5', 'PM10', 'SO2', 'NO2', 'CO', 'O3', 'TEMP', 'PRES', 'DEWP', 'RAIN', 'wd', 'WSP

# 示例用法
data_folder = 'C:\\Users\\shiye\\Desktop\\data'
analysis = DataAnalysis(data_folder)

# 时间分析示例
region = 'Aotizhongxin'
pollutant_type = 'PM2.5'
pollutant_data = analysis.pollutant_over_time(region, pollutant_type)
print(pollutant_data)

# 空间分析示例
time_point = pd.to_datetime('2014-05-01 00:00:00')
air_quality_distribution = analysis.air_quality_distribution(time_point)
print(air_quality_distribution)
```

```
[35064 rows x 5 columns]
              station  PM2.5   PM10   SO2    NO2  ...    PRES  DEWP  RAIN   wd  WSPM
10224     Aotizhongxin  101.0  160.0  15.0   78.0  ...  1002.3  13.2   0.0   SW   1.5
45288       Changping   90.0  136.0  12.0   44.0  ...   998.9   9.9   0.0    N   0.2
80352        Dingling   85.0   86.0   8.0   16.0  ...   998.9   9.9   0.0    N   0.2
115416         Dongsi   90.0  181.0  20.0   41.0  ...  1003.9  11.9   0.0  WSW   1.9
150480       Guanyuan  103.0  164.0  15.0   46.0  ...  1002.3  13.2   0.0   SW   1.5
185544        Gucheng  111.0  204.0   8.0  102.0  ...  1000.3  12.3   0.0  SSW   1.1
220608        Huairou  101.0  142.0   6.0   45.0  ...   999.5  11.0   0.0  ESE   1.0
255672   Nongzhanguan   93.0  156.0  17.0   49.0  ...  1003.9  11.9   0.0  WSW   1.9
290736         Shunyi   99.0  169.0  13.0   99.0  ...  1004.9  10.5   0.0  SSE   1.0
325800        Tiantan  100.0  137.0  14.0   35.0  ...  1003.9  11.9   0.0  WSW   1.9
360864         Wanliu  101.0  176.0  16.0   87.0  ...  1002.3  13.2   0.0   SW   1.5
```



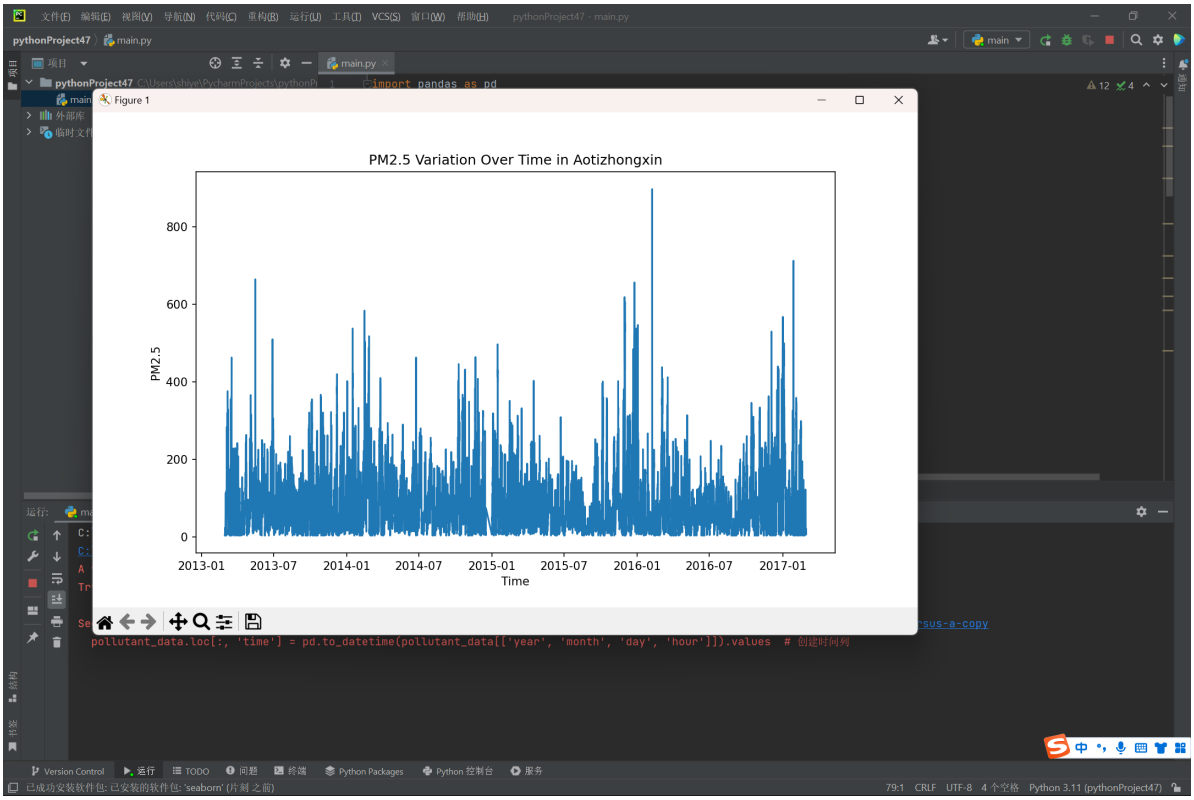


Figure 1
PM2.5 Variation Over Time in Aotizhongxin
PM2.5
Time
0
200
400
600
800
2013-01
2013-07
2014-01
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01

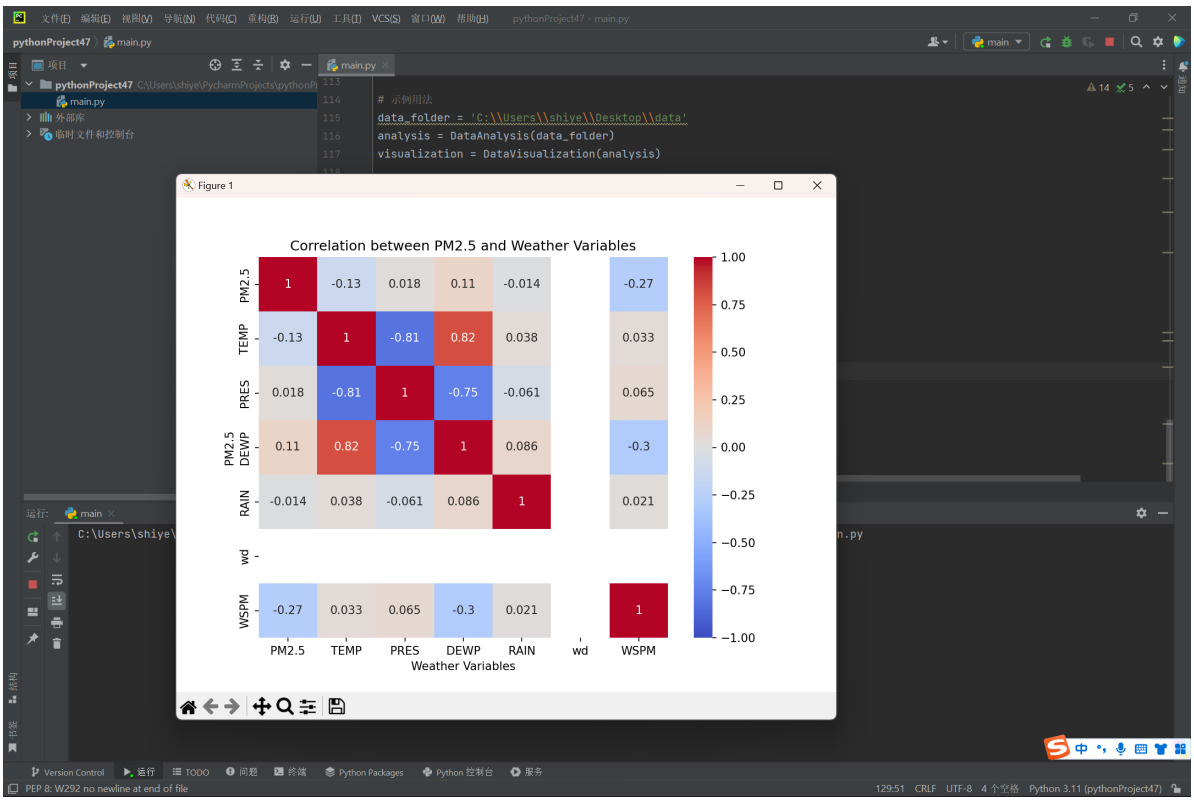


Figure 1
Correlation between PM2.5 and Weather Variables
PM2.5
TEMP
PRES
DEWP
RAIN
wd
WSPM
Weather Variables
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00