程设第六次作业

20377383 樊思涵

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。

初始化数据分析类并获得所有目标文件与其区域名称

```
class Data_process:
    def __init__(self,path):
        self.path=path
        self.filename_list = glob.glob(os.path.join(self.path,'*'+'.csv'))  #获得目标路径下的所有.csv文件
        #print(self.filename_list)
        self.station_list=[]
        for i in range(len(self.filename_list)):
            self.station_list.append(self.filename_list[i].split('_')[6])
        #print(self.station_list)
```

结果展示如下

```
['Aotizhongxin', 'Changping', 'Dingling', 'Dongsi', 'Guanyuan', 'Gucheng', 'Huairou', 'Nongzhanguan', 'Shunyi', 'Tiantan', 'Wanliu', 'Wanshouxigong']
```

内部读取文件+初始化的方法（返回 DataFrame）

初始化包括：规范格式，创建时间序列

```
54      def load_data_csv(self,filename):
55          """
56          读取单个路径下的.csv文件并做初始化
57          """
58          dic_dtype={'No':int,'year':int,'month':int,'day':int,
59          'hour':int,'PM2.5':float,'PM10':float,'SO2':float,
60          'NO2':float,'CO':float,'O3':float,'TEMP':float,
61          'PRES':float,'DEWP':float,'RAIN':float,'wd':str,
62          'WSPM':float,'station':str}
63          df = pd.read_csv(filename,header = 0,dtype=dic_dtype)
64          periods = pd.PeriodIndex(year=df["year"],month=df["month"],
65                                   day=df["day"],hour=df["hour"],freq="H")  #时间序列化
66          df1 = df.set_index(periods)
67          del_lis = ['No',"year","month","day","hour"]                     #删除无用列
68          for i in del_lis:
69              del df1[i]
70          return df1
71          #print(self.df_list)
```

结果展示如下：

```
                     PM2.5  PM10  SO2   NO2     CO    O3  TEMP    PRES  DEWP  RAIN   wd  WSPM station
2013-03-01 00:00       8.0   8.0  6.0  28.0  400.0  52.0  -0.7  1023.0 -18.8   0.0  NNW   4.4  Wanliu
2013-03-01 01:00       9.0   9.0  6.0  28.0  400.0  50.0  -1.1  1023.2 -18.2   0.0    N   4.7  Wanliu
2013-03-01 02:00       3.0   6.0  NaN  19.0  400.0  55.0  -1.1  1023.5 -18.2   0.0  NNW   5.6  Wanliu
2013-03-01 03:00      11.0  30.0  8.0  14.0    NaN   NaN  -1.4  1024.5 -19.4   0.0   NW   3.1  Wanliu
2013-03-01 04:00       3.0  13.0  9.0   NaN  300.0  54.0  -2.0  1025.2 -19.5   0.0    N   2.0  Wanliu
...                    ...   ...  ...   ...    ...   ...   ...     ...   ...   ...  ...   ...     ...
2017-02-28 19:00      11.0  27.0  4.0  20.0  300.0  81.0  12.6  1011.9 -14.3   0.0    N   2.0  Wanliu
2017-02-28 20:00      15.0  43.0  6.0  55.0  500.0  45.0   9.4  1012.3 -11.9   0.0  WSW   1.0  Wanliu
2017-02-28 21:00      13.0  35.0  7.0  48.0  500.0  48.0   8.7  1012.8 -13.7   0.0    N   1.1  Wanliu
2017-02-28 22:00      12.0  31.0  5.0  47.0  500.0  50.0   7.8  1012.9 -12.6   0.0  NNE   1.0  Wanliu
2017-02-28 23:00       7.0  25.0  6.0  86.0  700.0  11.0   7.0  1012.6 -11.2   0.0   NE   1.1  Wanliu

[35064 rows x 13 columns]
```

时间分析方法：

```
74      def time_analyze(self,station,type,mod = 'M'):
75          """
76          station为需要分析的区名
77          type为需要分析的数据名
78          mod为时间分析的模式 默认为以月为单位
79          输出对应模式对应排放量的数据分析
80          返回pandas数组用以可视化
81          """
82          for i in range(len(self.station_list)):
83              if self.station_list[i] == station:
84                  self.filename_time_analyze = self.filename_list[i]
85          #print(self.filename_time_analyze)
86          df = self.load_data_csv(self.filename_time_analyze)
87          #print(df)
88          df = df[type].resample(mod).mean()
89          #print(df,df.index)
90          for i in range(len(df)):
91              print("{}的平均{}排放量为{:.1f}".format(df.index[i],type,df[i]))
92          return df
```

结果显示：

```
2013-04的平均SO2排放量为24.7
2013-05的平均SO2排放量为30.4
2013-06的平均SO2排放量为14.6
2013-07的平均SO2排放量为9.3
2013-08的平均SO2排放量为6.9
2013-09的平均SO2排放量为13.8
2013-10的平均SO2排放量为22.4
2013-11的平均SO2排放量为26.0
2013-12的平均SO2排放量为43.9
2014-01的平均SO2排放量为58.5
2014-02的平均SO2排放量为56.4
2014-03的平均SO2排放量为36.6
2014-04的平均SO2排放量为20.0
2014-05的平均SO2排放量为15.5
2014-06的平均SO2排放量为6.3
2014-07的平均SO2排放量为4.5
2014-08的平均SO2排放量为4.0
2014-09的平均SO2排放量为5.5
2014-10的平均SO2排放量为10.4
2014-11的平均SO2排放量为17.2
2014-12的平均SO2排放量为25.2
2015-01的平均SO2排放量为38.0
2015-02的平均SO2排放量为30.4
2015-03的平均SO2排放量为22.8
2015-04的平均SO2排放量为9.7
2015-05的平均SO2排放量为8.7
2015-06的平均SO2排放量为7.3
2015-07的平均SO2排放量为5.7
2015-08的平均SO2排放量为4.1
2015-09的平均SO2排放量为5.8
2015-10的平均SO2排放量为6.4
2015-11的平均SO2排放量为10.6
2015-12的平均SO2排放量为23.6
2016-01的平均SO2排放量为22.8
2016-02的平均SO2排放量为18.2
2016-03的平均SO2排放量为22.3
2016-04的平均SO2排放量为11.4
2016-05的平均SO2排放量为9.4
2016-06的平均SO2排放量为5.5
2016-07的平均SO2排放量为3.5
2016-08的平均SO2排放量为2.7
2016-09的平均SO2排放量为3.0
2016-10的平均SO2排放量为3.3
2016-11的平均SO2排放量为9.8
2016-12的平均SO2排放量为18.0
2017-01的平均SO2排放量为20.4
2017-02的平均SO2排放量为21.3
```

空间分析方法：

```python
    def space_analyze(self,type):
        """
        type为需要分析的数据名
        输出对应
        返回pandas数组用以可视化
        """
        df_space = pd.DataFrame()
        df_space['station'] = self.station_list
        data_list=[]
        for i in range(len(self.station_list)):
            df = self.load_data_csv(self.filename_list[i])
            data_list.append(df[type].mean())
        df_space[type] = data_list
        df_space = df_space.set_index('station')
        print(df_space.index)
        for i in range(len(df_space)):
            print("{}近年的平均{}排放量为{:.1f}".format(df_space.index[i],type,df_space[type][i]))
        return df_space

```

结果展示：

```
Aotizhongxin近年的平均SO2排放量为17.4
Changping近年的平均SO2排放量为15.0
Dingling近年的平均SO2排放量为11.7
Dongsi近年的平均SO2排放量为18.5
Guanyuan近年的平均SO2排放量为17.6
Gucheng近年的平均SO2排放量为15.4
Huairou近年的平均SO2排放量为12.1
Nongzhanguan近年的平均SO2排放量为18.7
Shunyi近年的平均SO2排放量为13.6
Tiantan近年的平均SO2排放量为14.4
Wanliu近年的平均SO2排放量为18.4
Wanshouxigong近年的平均SO2排放量为17.1
```

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。

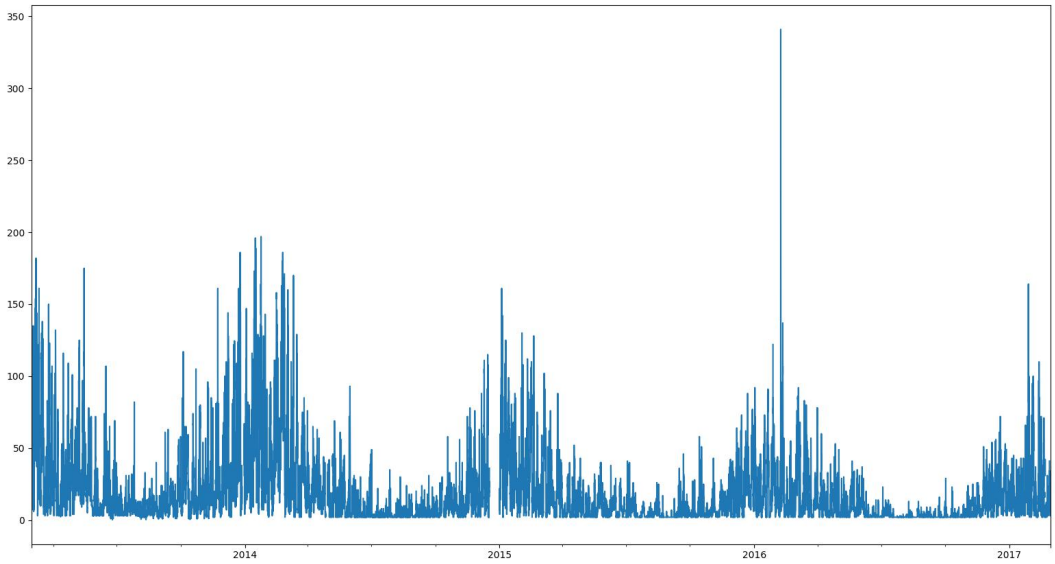
分析结果可视化

```python
class Data_view:
    def  __init__(self,data_time=[],data_space=[]):
        """
        将需要的时间与空间数据做初始化
        """
        self.data_time = data_time
        self.data_space = data_space

    def time_view(self):
        """
        画出对应的时间分布图
        """
        self.data_time.plot(subplots=True, figsize=(10,12))
        plt.show()

    def space_view(self):
        """
        画出对应的空间分布图
        """
        self.data_space.plot(kind = 'bar')
        plt.xticks(rotation = 360)
        plt.show()
        self.data_space.plot(kind = 'pie',subplots=True,autopct='%.2f%%')
        plt.show()

```

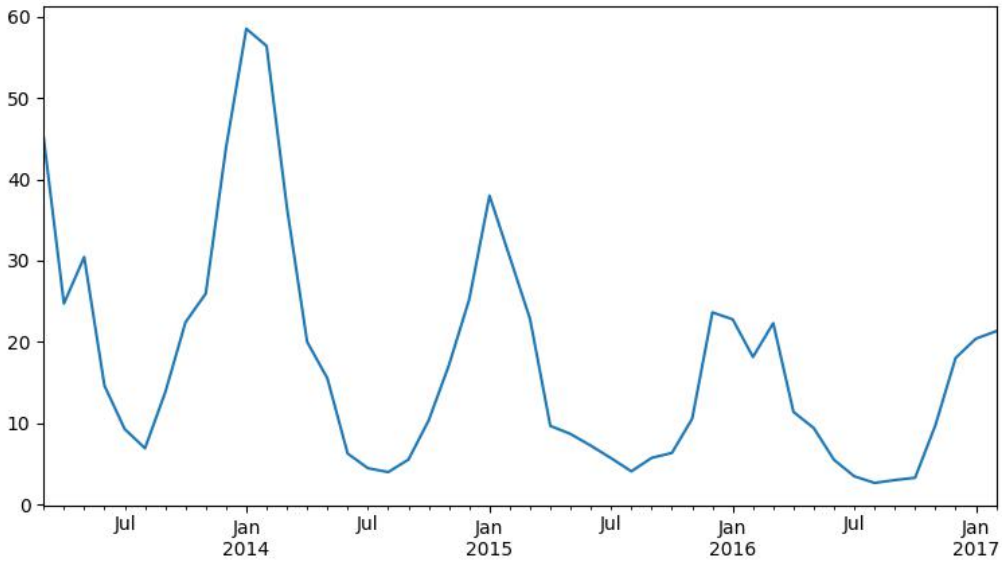
结果展示（以 Aotizhongxin 为例）

首先看看以小时为单位（原始格式）的数据可视化：

（SO2 排放量为例）

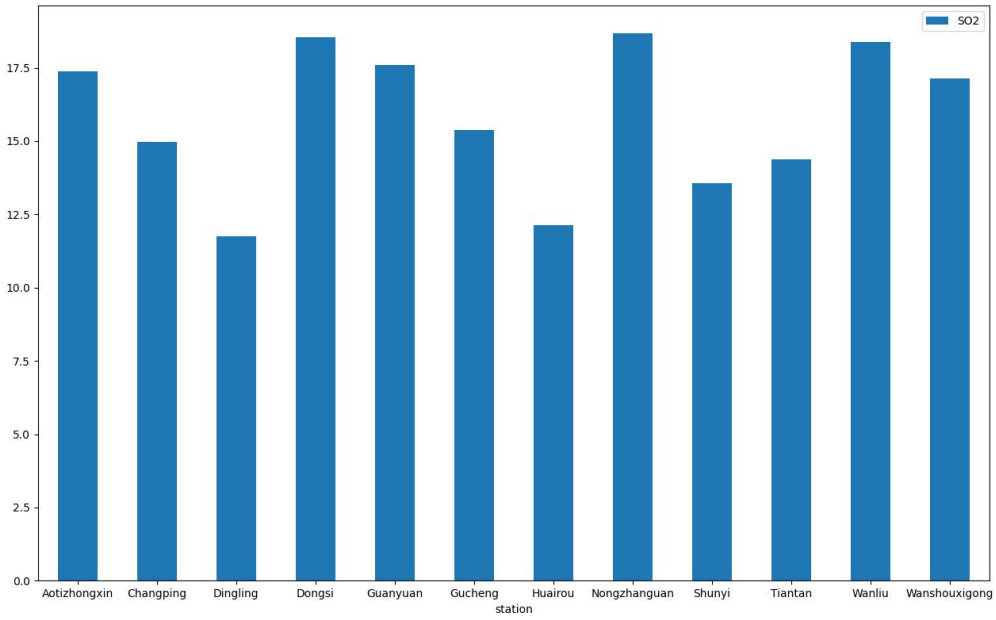


可视化效果并不理想，考虑以月为单位：

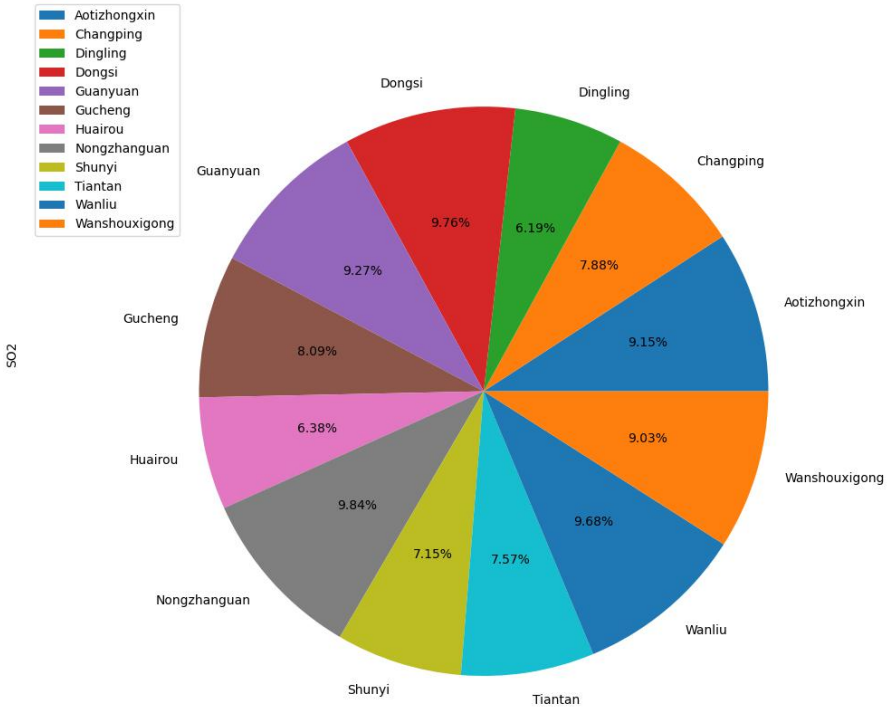


此时可以明显看出排放量随季度的变化和年与年的变化

空间分析柱状图为



空间分析饼状图为



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。

实现 Error 类

```python
class DataNotNumError(ValueError):
    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 = "{} {}-{}-{}-{} {} is not a valid number.".format(region,year,month,day,hour,pollutant)

```

在数据分析类中实现 examine 方法

```python
    def examine(self):
        """
        应在load前使用以确保数据形式正确
        """
        for i in self.filename_list:
            data = pd.read_csv(i,header = 0)
            x,y=map(list,np.where(data.isnull()))          #获得异常数据位置
            if len(x) > 0:
                for j in range(len(x)):
                    try:
                        pollutant = data.columns.values[y[j]]
                        year = data['year'][x[j]]
                        region = "Aotizhongxin"
                        month = data['month'][x[j]]
                        day = data['day'][x[j]]
                        hour = data['hour'][x[j]]
                        raise DataNotNumError(region, year, month, day, hour,pollutant)
                    except DataNotNumError as error:
                        print(error.message)
                    #break
                data.fillna(method='pad', inplace=True)    #将数据按前一行补齐
                print(np.any(data.isnull()))
            #break
```

结果展示

先不进行 except 异常捕获

```
PS E:\code\py_code> python -u "e:\code\py_code\week7\week7.py"
Traceback (most recent call last):
  File "e:\code\py_code\week7\week7.py", line 148, in <module>
    main()
  File "e:\code\py_code\week7\week7.py", line 139, in main
    Dp.examine()
  File "e:\code\py_code\week7\week7.py", line 45, in examine
    raise DataNotNumError(region, year, month, day, hour,pollutant)
__main__.DataNotNumError: ('Aotizhongxin', 2013, 3, 4, 2, 'SO2')
PS E:\code\py_code>
```

进行 except 异常捕获

（示例仅输出一条异常就 break）

```
PS E:\code\py_code> python -u "e:\code\py_code\week7\week7.py"
Aotizhongxin 2013-3-4-2 SO2 is not a valid number.
异常数据处理后异常状态为：（False即为无异常）
False
```

异常处理成功

以下为主函数代码

```python
def main():
    Dp = Data_process(path)
    Dp.examine()
    dt = Dp.time_analyze(station = 'Aotizhongxin',type = 'SO2')
    ds = Dp.space_analyze(type = 'SO2')
    Dv=Data_view(data_time=dt,data_space=ds)
    Dv.time_view()
    Dv.space_view()


if __name__ == '__main__':
    main()
```