Week7

19377419 孙启航

作业描述

> 经济管理中通常有大量的数据以excel格式存在，如本次作业提供的中国各省长周期CO2排放数据
>
> 数据格式如下：所有数据按年份存储在不同的文件中,Province_sectoral_CO2_emissions_20xx.xlsx，其中20xx为年份。单个excel文件中，"sum"数据页给出了各省的总CO_2排放量以及来源明细，其余以省命名的数据页则给出了各省不同行业不同来源的CO_2排放量。
>
> 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（如某区域某类型排放随时间的变化）和空间分析（某一年全国排放的空间分布态势）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、饼等形式对结果进行呈现。
> 3. 由于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。因此需要实现NotNumError类，继承ValueError，并加入新属性year，province，industry，type，对数据进行检测，若取到的一列数据中包含nan，则抛出该异常，并提供异常相关的年份，省份，工业和排放类型等信息。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. 由于部分省份排放总量数据为0，要求在计算比例时进行检验，若检验发现总量为0，则抛出ZeroDivisionError，并打印对应的行名等信息。
> 5. 按时间分析时，注意观察不同区域随时间排放量的变化，是否存在一些明显的趋势，以及趋势的空间差异，并思考这些趋势及差异的管理意义与政策启发。

Error.py 实现NotNumError类等

```
class NotNumError(ValueError):
    '''
    检测到数据表中的空值时，抛出该异常
    并保存该数据对应的年度，省份，部门，排放类型等信息
    '''
    def __init__(self, year, province, industry, type):
        self.year = year
        self.province = province
        self.industry = industry
        self.type = type
        self.message = f"Not number error: \nYear: {self.year}\nProvince:
{self.province}\nIndustry:{self.industry}\nType: {self.type}\n"


class FindNoFilesError(Exception):
    '''
    没有查找到给定目录下的xlsx文件时，抛出该异常
    '''
    def __init__(self, dir):
        self.dir = dir
        self.message = f"Could not find xlsx file in current directory:
{self.dir}"
```

```python
class ProvinceParameterError(Exception):
    '''
    数据表中没有对应的省份名时，抛出该异常（说明省份名称输入出现错误）
    '''
    def __init__(self, province):
        self.message = f"Area parameter error: {province} doesn't exist."


class IndustryParameterError(Exception):
    '''
    数据表中没有对应的部门名时，抛出该异常（说明部门名称输入出现错误）
    '''
    def __init__(self, industry):
        self.message = f"Industry parameter error: {industry} doesn't exist"


class TypeParameterError(Exception):
    '''
    数据表中没有对应的排放类型时，抛出该异常（说明排放类型名称输入出现错误）
    '''
    def __init__(self, type):
        self.message = f"Type parameter error: {type} doesn't exist."


class TimeParameterError(Exception):
    '''
    没有对应年份的数据时，抛出该异常（说明时间的输入出现错误）
    '''
    def __init__(self, time):
        self.message = f"Time parameter error: {time} doesn't exist."


class NotSpatialDataError(Exception):
    '''
    在绘图时如果传入的数据并不是空间分析后的数据，抛出该异常
    '''
    def __init__(self):
        self.message = f"The data is not spatial data."


class NotTimeDataError(Exception):
    '''
    在绘图时如果传入的数据并不是时间分析后的数据，抛出该异常
    '''
    def __init__(self):
        self.message = f"The data is not time data."


class ZeroDivisionError(Exception):
    '''
    如果某个省市的Sum列对应的数据为0，那么在之后计算各部分的排放比例时已经会出现
DivisionByZero异常，提前进行检测
    '''
    def __init__(self, province, time, type):
        self.province = province
        self.time = time
        self.type = type
        self.message = f"{self.type} CO2 emission of {self.province} in
{self.time} is 0."
```

visualization.py 实现数据可视化类

import & 全局变量定义

```
from Error import NotTimeDataError
from Error import NotSpatialDataError, NotTimeDataError
from pyecharts import Map, Timeline
import matplotlib.pyplot as plt
import sys

namemap = {"Beijing": "北京", "Tianjin": "天津", "Hebei": "河北", "Shanxi": "山西",
"InnerMongolia": "内蒙古", "Liaoning": "辽宁", "Jilin": "吉林", "Shanghai": "上海",
"Heilongjiang": "黑龙江", "Jiangsu": "江苏", "Zhejiang": "浙江", "Anhui": "安徽",
"Fujian": "福建", "Jiangxi": "江西","Shandong": "山东", "Henan": "河南", "Hubei":
"湖北", "Hunan": "湖南", "Guangdong": "广东", "Guangxi": "广西", "Hainan": "海南",
"Chongqing": "重庆", "Sichuan": "四川", "Guizhou": "贵州", "Yunnan": "云南",
"Shaanxi": "陕西", "Gansu": "甘肃", "Qinghai": "青海", "Ningxia": "宁夏",
"Xinjiang": "新疆"}
# 因为数据表中的省份名称为英文名称，但是pyecharts包绘制地图时要求中文名称，所以建立中英文省市
名的映射关系方便后续的分析
```

类和类方法定义

绘制折线图（某省市/某年度）

```
class Visualization(object):

    def line_graph(dic):
        # 根据时间分析或者空间分析后的数据绘制折线图
        x = list(dic["Value"].keys())
        y = list(dic["Value"].values())

        if "Province" in dic.keys():

            plt.plot(x, y, linewidth=2, marker='*',
                     label="The emission of CO2")
            plt.tick_params(axis='both', labelsize=6)
            plt.title("Time analysis of %s CO2 emission in %s (1997 - 2015)" %
                      (dic["Type"], dic["Province"]), fontsize=10)
            plt.legend(loc='upper left')
            plt.show()

        elif "Time" in dic.keys():
            x = x[:-2]
            y = y[:-2]
            # plt.plot(x, y, linewidth=2, marker='*', label="The emission of
CO2", color="#0A2463", alpha=0.5)
```

```
            plt.bar(x, y, width=0.9, color="#0A2463", alpha=0.6)
            plt.xticks(x, x, rotation=55, fontsize=5)
            plt.yticks(fontsize=8)

            plt.title("Spatial analysis of %s CO2 emission of Year %s" %
                      (dic["Type"], dic["Time"]), fontsize=10)
            plt.show()
```

绘制饼图（某年度）

```
def pie_graph(dic):
        '''
        根据空间分析后的数据绘制饼状图
        '''
        x = list(dic["Value"].keys())
        y = list(dic["Value"].values())
        try:
            if "Time" in dic.keys():
                x = x[:-2]
                y = y[:-2]

                colors = ("#ECCBD9", "#E1EFF6", "#97D2FB", "#83BCFF", "#80FFE8",
                          "#4B4A67", "#2364A4", "#FEC601", "#EA7317", "#AF42AE")
                plt.figure(figsize=(8, 12), dpi=80)
                patches, l_text = plt.pie(
                    y, radius=3, autopct=None, colors=colors, labels=x,
labeldistance=1.03)
                for i in l_text:
                    i.set_size(6)
                plt.title("%s proportion of provinces in %s" %
                          (dic["Type"], dic["Time"]), fontsize=14)
                plt.axis("equal")
                plt.show()

            elif "Province" in dic.keys():
                raise NotSpatialDataError()
        except NotSpatialDataError as NSDE:
            print(NSDE.message)
            sys.exit(0)
```

绘制空间分析地图（某年度）

```
def province_distribution(dic):
        '''
        利用Pyecharts，对进行空间分析后的数据绘制地图'''
        try:
            if "Time" in dic.keys():

                if "Industry" in dic.keys():
                    temp = list(dic["Value"].keys())
                    values = list(dic["Value"].values())
                    map = Map("CO2 emission of China",
"Time:%s\nType:%s\nIndustry:%s" %
                              (dic["Time"], dic["Type"], dic["Industry"]),
width=1200, height=800)
                    province = []
                    for i in temp:
                        province.append(namemap[i])
                    map.add('', province, values, maptype='china',
                            visual_range=[0, max(values)], is_visualmap=True)

                    map.render(path="BUAA_21/Week7/China CO2 emission (%s-%s-
%s).html" %
                               (dic["Time"], dic["Type"], dic["Industry"]))
                else:
                    temp = list(dic["Value"].keys())[:-2]
                    values = list(dic["Value"].values())[:-2]

                    map = Map("CO2 emission of China", "Time:%s\nType:%s\n" % (
                        dic["Time"], dic["Type"]), width=1200, height=800)
                    province = []
                    for i in temp:
                        province.append(namemap[i])
                    map.add('', province, values, maptype='china',
                            visual_range=[0, max(values)], is_visualmap=True)

                    map.render(path="BUAA_21/Week7/China CO2 emission (%s-
%s).html" %
                               (dic["Time"], dic["Type"]))
            else:
                raise NotSpatialDataError()
        except NotSpatialDataError as NSDE:
            print(NSDE.message)
            sys.exit(0)
```

绘制时间轴上的地图（某省）

```
    def province_time_line(dic):
        '''
        利用Pyecharts对进行时间分析后的数据进行分析，绘制时间轴地图'''

        try:
            if "Province" in dic.keys():
                if "Industry" in dic.keys():
                    timedata = Timeline(
                        is_auto_play=True, timeline_bottom=0)
                    max_num = max(dic["Value"].values())
                    for year, data in dic["Value"].items():
                        map = Map("CO2 emission of China",
"Province:%s\nType:%s\nIndustry:%s" %
                                  (dic["Province"], dic["Type"],
dic["Industry"]), width=1200, height=800)

                        ma = map.add('', [namemap[dic["Province"]]], [
                                     data], maptype="china", visual_range=[0,
max_num], is_visualmap=True)
                        timedata.add(time_point=year, chart=ma)
                    timedata.render(path="BUAA_21/Week7/1997-2015 CO2 emission
(%s-%s-%s).html" %
                                    (dic["Province"], dic["Type"],
dic["Industry"]))
                else:
                    timedata = Timeline(is_auto_play=True, timeline_bottom=0)
                    max_num = max(dic["Value"].values())
                    for year, data in dic["Value"].items():
                        map = Map("CO2 emission of China", "%s-%s" %
                              (dic["Province"], dic["Type"]), width=1200,
height=800)

                        ma = map.add('', [namemap[dic["Province"]]], [
                            data], maptype="china", visual_range=[0, max_num],
is_visualmap=True)
                        timedata.add(time_point=year, chart=ma)
                    timedata.render(path="BUAA_21/Week7/1997-2015 CO2 emission
(%s-%s).html" %
                               (dic["Province"], dic["Type"]))
            else:
                raise NotTimeDataError

        except NotTimeDataError as NTDE:
            print(NTDE.message)
            sys.exit(0)
```

绘制时间轴上的地图（全国范围）

```
    def time_spatial_visualization(dic):

        time_range = list(dic["Value"].keys())
        timedata = Timeline(is_auto_play = True,timeline_bottom = 0)
        for time in time_range:
            temp = list(dic["Value"][time].keys())
            values = list(dic["Value"][time].values())
            province = []
            for i in temp:
                province.append(namemap[i])
            map = Map("CO2 emission of China", "Time: %s\nType: %s\nIndustry:
%s\n"%(time,dic["Type"],dic["Industry"]),width=1200, height=800)
            ma = map.add('',province,values,maptype = "china",visual_range =
[0,max(values)],is_visualmap = True)
            timedata.add(time_point=time, chart=ma)
        timedata.render(path = "BUAA_21/Week7/1997-2015 CO2 emission (China-%s-
%s).html"%(dic["Type"],dic["Industry"]))
```

main.py 实现数据分析类

Import环节

```
from Error import NotNumError, IndustryParameterError, FindNoFilesError,
ProvinceParameterError, TimeParameterError, TypeParameterError,
 ZeroDivisionError
from visualization import Visualization
import xlrd
import os
import sys
```

类和类方法定义

定义名为ReadData的数据分析类

利用os.walk方法查找目录下xlsx文件，同时将每个文件名中的时间进行读取保存在类属性time_range的列表中**（当然前提是已知了文件名的格式，最后4位数字即数据的年份）**

利用xlrd包进行excel的读取，并将打开后的对象保存在类属性xlsx_list中

```
class ReadData(object):
    '''
    数据分析类，实现数据读取以及基本的时间空间分析'''

    def __init__(self):
        self.file_list = []
        self.xlsx_list = []

    def find_xlsx(self, dir):
        '''
        找到该目录下的所有xlsx文件，保存到self.file_list中去
        '''
        self.time_range = []
        file_list = []
        for root_dir, sub_dir, files in os.walk(r''+dir):
            for file in files:
                if file.endswith('.xlsx'):
                    file_list.append(os.path.join(root_dir, file))
        try:
            if file_list == []:
                raise FindNoFilesError(dir)
        except FindNoFilesError as FNFE:
            print(FNFE.message)
            sys.exit(0)

        self.file_list = file_list
        for i in self.file_list:
            self.time_range.append(i[-9:-5])

    def read(self):
        '''
        对所有xlsx进行读取，写入到列表中去
        '''

        for i in self.file_list:
            self.xlsx_list.append(xlrd.open_workbook(i))
```

之后在定义的六个函数中分别对数据进行不同形式的分析，并将结果以字典形式保存在对应的类属性中

（之后的部分仍是类ReadData中的定义）

```
def time_analysis(self, province, type):
        '''
        对某个省的整个时间范围内的数据进行分析
        提取每个年份下的sum表，读取某个省份某个排放类型的数据
        参数province和type以外部参数传入
        将结果以字典形式保存在类的time_analysis_dict中
        '''
        self.time_analysis_dict = {}

        time_analysis_value_list = []
        for xlsx in self.xlsx_list:
            sheet1 = xlsx.sheets()[0]
            try:
                if province not in sheet1.col_values(0):
                    raise ProvinceParameterError(province)
            except ProvinceParameterError as PPE:
                print(PPE.message)
                sys.exit(0)
            finally:
                pass

            province_index = sheet1.col_values(0).index(province)

            try:
                if type not in sheet1.row_values(0):
                    raise TypeParameterError(type)
            except TypeParameterError as TPE:
                print(TPE.message)
            finally:
                pass

            type_index = sheet1.row_values(0).index(type)
            time_analysis_value_list.append(sheet1.cell(province_index,
type_index).value)

        self.time_analysis_dict["Value"] = dict(
            zip(self.time_range, time_analysis_value_list))
        self.time_analysis_dict["Type"] = type
        self.time_analysis_dict["Province"] = province
```

```
    def spatial_analysis(self, time, type):
        '''
        对全国范围内某年的数据进行分析
        提取每个年份下的sum表，读取某个年份某个排放类型的数据
        参数time和type以外部参数传入
        将结果以字典形式保存在类的spatial_analysis属性中
        '''
        self.spatial_analysis_dict = {}
        try:
            if time not in self.time_range:
                raise TimeParameterError(time)
        except TimeParameterError as TPE:
            print(TPE.message)
            sys.exit(0)
        finally:
            pass

        time_index = self.time_range.index(time)
        xlsx = self.xlsx_list[time_index]
        sheet1 = xlsx.sheets()[0]

        try:

            if type not in sheet1.row_values(0):
                raise TypeParameterError(type)
        except TypeParameterError as TPE:
            print(TPE.message)
            sys.exit(0)
        type_index = sheet1.row_values(0).index(type)

        col_value = sheet1.col_values(type_index)[1:]
        provinces = sheet1.col_values(0)[1:]
        col_value.pop(-2)
        provinces.pop(-2)
        self.spatial_analysis_dict["Value"] = dict(zip(provinces, col_value))
        self.spatial_analysis_dict["Type"] = type
        self.spatial_analysis_dict["Time"] = time
```

```
    def proportion_analysis(self):
        '''
        对excel文件中名为sum的sheet进行分析

        如果一个省的Total数据为0，那么在计算分部门碳排放比例时，一定会出现Division By zero
错误
        所以要对Total数据为0的数据进行记录，保存其年份，部门，省份等
        '''
        try:
            for i in self.time_range:
                self.spatial_analysis(i, "Total")
                value_dict = self.spatial_analysis_dict["Value"]
                sum_num = value_dict["Sum-CO2"]
                try:
                    for i in value_dict.keys():
                        if i != "Sum-CO2" and (value_dict[i] != 0):
                            # print(value_dict[i]/sum_num)
                            # print("The total emission of %s in %s is %.2f" %
(i, self.spatial_analysis_dict["Time"], value_dict[i]))
                            pass
                        elif i == "Sum-CO2":
                            continue
                        elif value_dict[i] == 0:
                            raise ZeroDivisionError(
                                i, self.spatial_analysis_dict["Time"],
self.spatial_analysis_dict["Type"])
                except ZeroDivisionError as ZDE:
                    print("\n"+ZDE.message+"\n")
        except AttributeError as AE:
            print(AE)
            sys.exit(0)
```

```
    def detailed_time_analysis(self, province, type, industry):
        '''
        详细的时间分析，不针对Sum表，而是针对更加细致的分省分部门的数据表进行分析
        对整个时间范围内，某个省某个部分的某个排放类型的排放数据进行分析
        将分析后的结果以字典形式保存在类的detailed_time_analysis_dict属性中
        '''
        self.detailed_time_analysis_dict = {}
        time_range = []
        detail_time_analysis_list = []
        year_index = 0
        for i in self.xlsx_list:
            sheet = i.sheet_by_name(province)
            try:
                if type not in sheet.row_values(0):
                    raise TypeParameterError(type)
                elif industry not in sheet.col_values(0):
                    raise IndustryParameterError(industry)

            except TypeParameterError as TPE:
                print(TPE.message)
                sys.exit(0)
            except IndustryParameterError as IPE:
                print(IPE.message)
                sys.exit(0)
            finally:
                pass

            type_index = sheet.row_values(0).index(type)
            industry_index = sheet.col_values(0).index(industry)
            cell_value = sheet.cell(industry_index, type_index).value

            try:
                if cell_value != '':
                    detail_time_analysis_list.append(cell_value)
                    time_range.append(self.time_range[year_index])
                    year_index += 1
                else:
                    year_index += 1
                    raise NotNumError(
                        year=self.time_range[year_index-1], province=province,
industry=industry, type=type)
            except NotNumError as NNE:
                print(NNE.message)
            finally:
                pass

        self.detailed_time_analysis_dict["Value"] = dict(
            zip(time_range, detail_time_analysis_list))
        self.detailed_time_analysis_dict["Type"] = type
        self.detailed_time_analysis_dict["Industry"] = industry
        self.detailed_time_analysis_dict["Province"] = province
```

```
    def detailed_spatial_analysis(self, time, type, industry):
        '''
        详细的空间分析，不针对Sum表，而是针对更加细致的分省分部门的数据表进行分析
        对某一年，全国范围内某个部门的某个排放类型的排放数据进行分析
        将分析后的结果以字典形式保存在类的detailed_spatial_analysis_dict属性中
        '''
        self.detailed_spatial_analysis_dict = {}
        self.detailed_spatial_analysis_dict["Value"] = {}
        # print(self.time_range)
        time_index = self.time_range.index(str(time))
        xlsx = self.xlsx_list[time_index]
        sheet0 = xlsx.sheets()[0]

        province_lis = sheet0.col_values(0)[1:-2]

        for province in province_lis:
            sheet = xlsx.sheet_by_name(province)
            try:
                if type not in sheet.row_values(0):
                    raise TypeParameterError(type)
                elif industry not in sheet.col_values(0):
                    raise IndustryParameterError(industry)
            except TypeParameterError as TPE:
                print(TPE.message)
                sys.exit(0)
            except IndustryParameterError as IPE:
                print(IPE.message)
                sys.exit(0)
            finally:
                pass

            type_index = sheet.row_values(0).index(type)
            industry_index = sheet.col_values(0).index(industry)
            cell_value = sheet.cell(industry_index, type_index).value
            try:
                if cell_value != '':
                    self.detailed_spatial_analysis_dict["Value"][province] =
cell_value
                else:
                    raise NotNumError(
                        year=time, province=province, industry=industry,
type=type)
            except NotNumError as NNE:
                print(NNE.message)

        self.detailed_spatial_analysis_dict["Time"] = time
        self.detailed_spatial_analysis_dict["Type"] = type
        self.detailed_spatial_analysis_dict["Industry"] = industry
```

```
    def time_and_spatial_analysis(self, type, industry):
        '''
        整体全面的分析，对整个时间范围内，全国各个省某个部门某排放类型的数据
        将分析结果以字典形式保存在类的time_and_spatial_analysis属性中'''
        self.time_and_spatial_analysis_dic = {}
        self.time_and_spatial_analysis_dic["Value"] = {}

        province_lis = self.xlsx_list[0].sheets()[0].col_values(0)[1:-2]

        tot = 0
        for i in self.xlsx_list:
            time = self.time_range[tot]
            tot += 1
            self.time_and_spatial_analysis_dic["Value"][time] = {}
            for province in province_lis:
                sheet = i.sheet_by_name(province)
                try:
                    if type not in sheet.row_values(0):
                        raise TypeParameterError(type)
                    elif industry not in sheet.col_values(0):
                        raise IndustryParameterError(industry)
                except TypeParameterError as TPE:
                    print(TPE.message)
                    sys.exit(0)
                except IndustryParameterError as IPE:
                    print(IPE.message)
                    sys.exit(0)
                finally:
                    pass
                type_index = sheet.row_values(0).index(type)
                industry_index = sheet.col_values(0).index(industry)
                cell_value = sheet.cell(industry_index, type_index).value
                try:
                    if cell_value != '':
                        self.time_and_spatial_analysis_dic["Value"][time]
[province] = cell_value
                    else:
                        raise NotNumError(
                            year=time, province=province, industry=industry,
type=type)
                except NotNumError as NNE:
                    print(NNE.message)

        self.time_and_spatial_analysis_dic["Type"] = type
        self.time_and_spatial_analysis_dic["Industry"] = industry
```

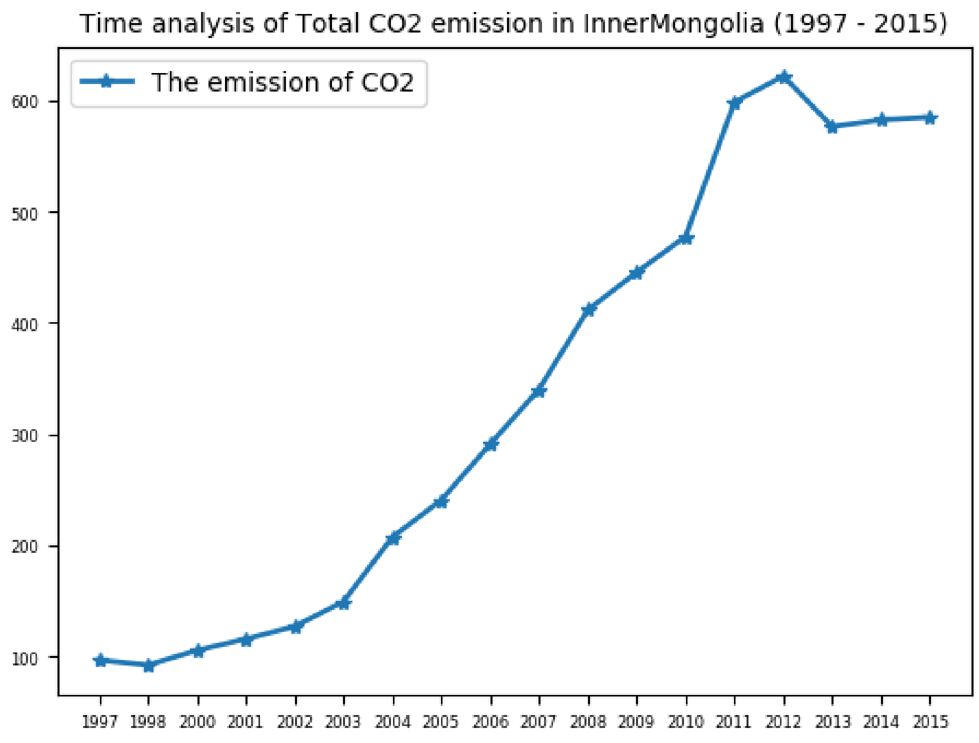
main函数

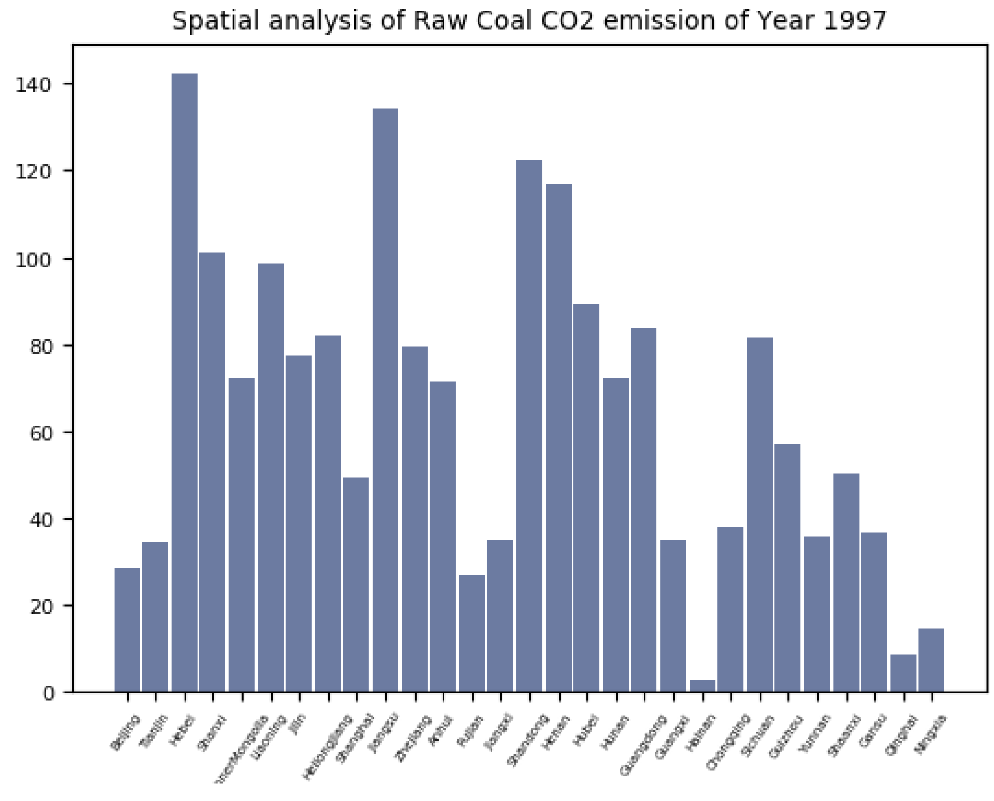
```
if __name__ == '__main__':
    class_read_data = ReadData()
    class_read_data.find_xlsx("BUAA_21/Week7/CO2")
    class_read_data.read()

    class_read_data.time_analysis("InnerMongolia", "Total")
    Visualization.line_graph(class_read_data.time_analysis_dict)

    class_read_data.spatial_analysis("1997", "Raw Coal")
    Visualization.line_graph(class_read_data.spatial_analysis_dict)

    Visualization.pie_graph(class_read_data.spatial_analysis_dict)

    Visualization.province_time_line(class_read_data.time_analysis_dict)

    Visualization.province_distribution(class_read_data.spatial_analysis_dict)

    class_read_data.detailed_time_analysis(province = "InnerMongolia",type =
"Raw Coal", industry= "Total Consumption")

 Visualization.province_time_line(class_read_data.detailed_time_analysis_dict)

    class_read_data.detailed_spatial_analysis(time = 2001, type = "Total",
industry = "Total Consumption")

 Visualization.province_distribution(class_read_data.detailed_spatial_analysis_d
ict)

    class_read_data.time_and_spatial_analysis(type="Total", industry="Total
Consumption")

 Visualization.time_spatial_visualization(class_read_data.time_and_spatial_analy
sis_dic)
```

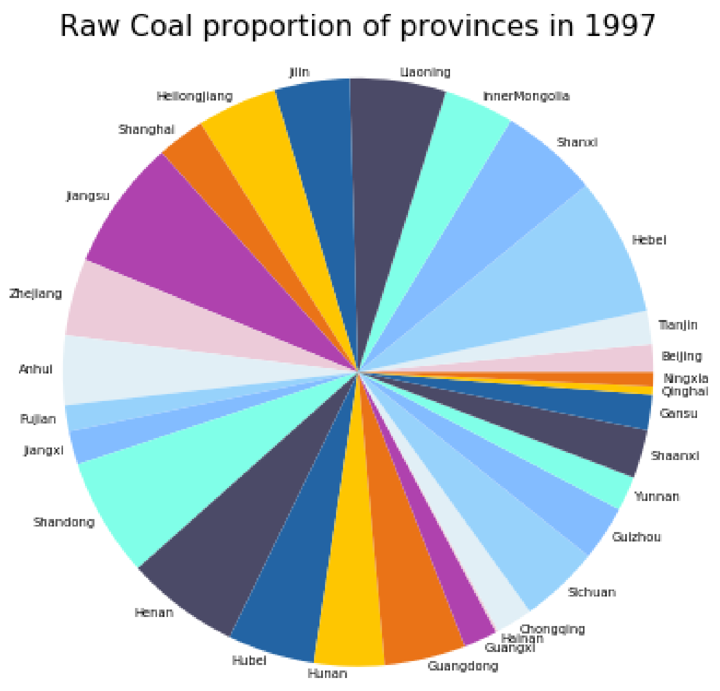
```
    class_read_data.time_and_spatial_analysis(type="Raw Coal", industry="Total
Consumption")

 Visualization.time_spatial_visualization(class_read_data.time_and_spatial_analy
sis_dic)

    class_read_data.proportion_analysis()
```

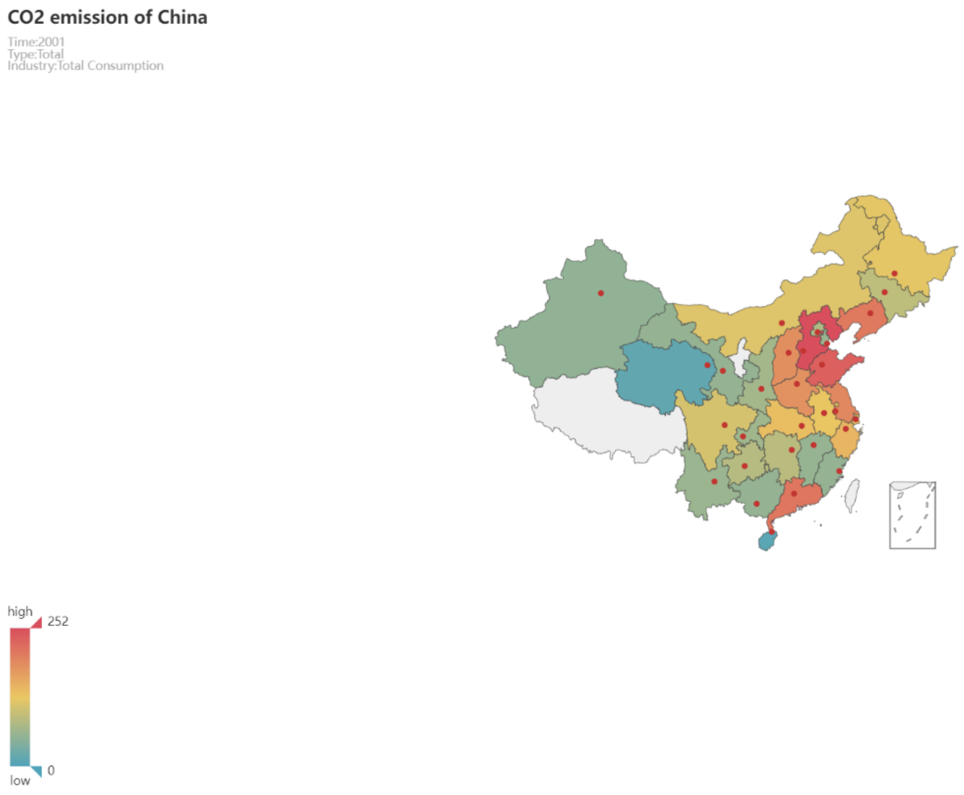
运行结果

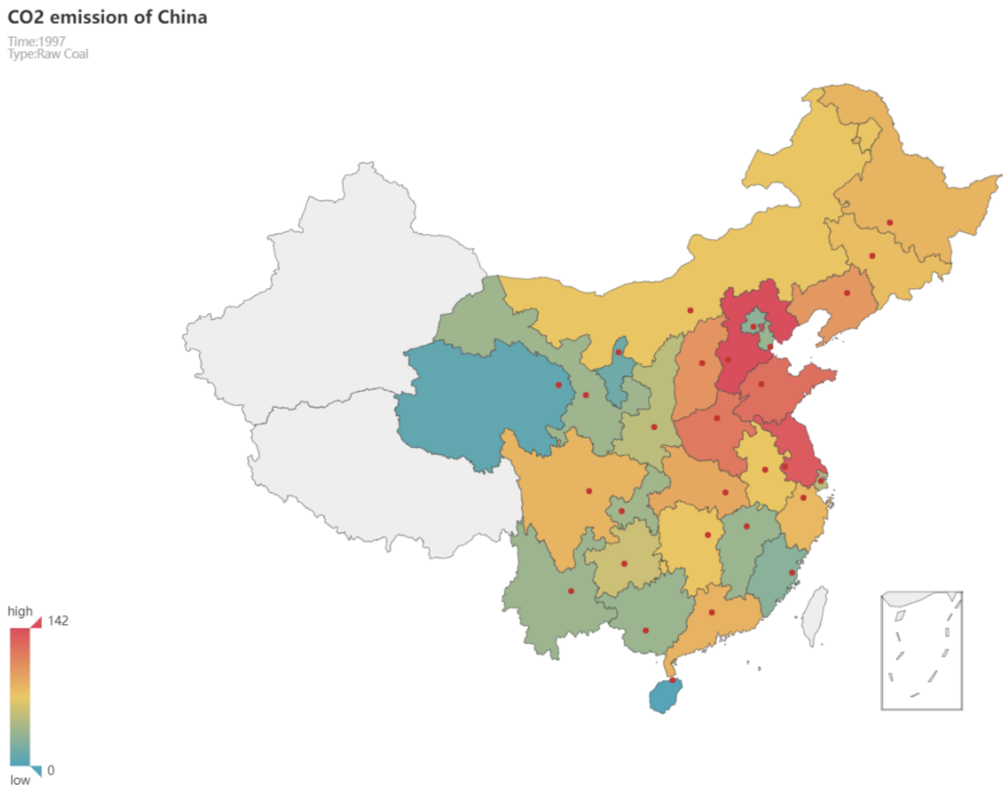
数据可视化结果（部分）




Spatial analysis of Raw Coal CO2 emission of Year 1997
0
20
40
60
80
100
120
140
Beijing
Tianjin
Hebei
Shanxi
InnerMongolia
Liaoning
Jilin
Heilongjiang
Shanghai
Jiangsu
Zhejiang
Anhui
Fujian
Jiangxi
Shandong
Henan
Hubei
Hunan
Guangdong
Guangxi
Hainan
Chongqing
Sichuan
Guizhou
Yunnan
Shaanxi
Gansu
Qinghai
Ningxia

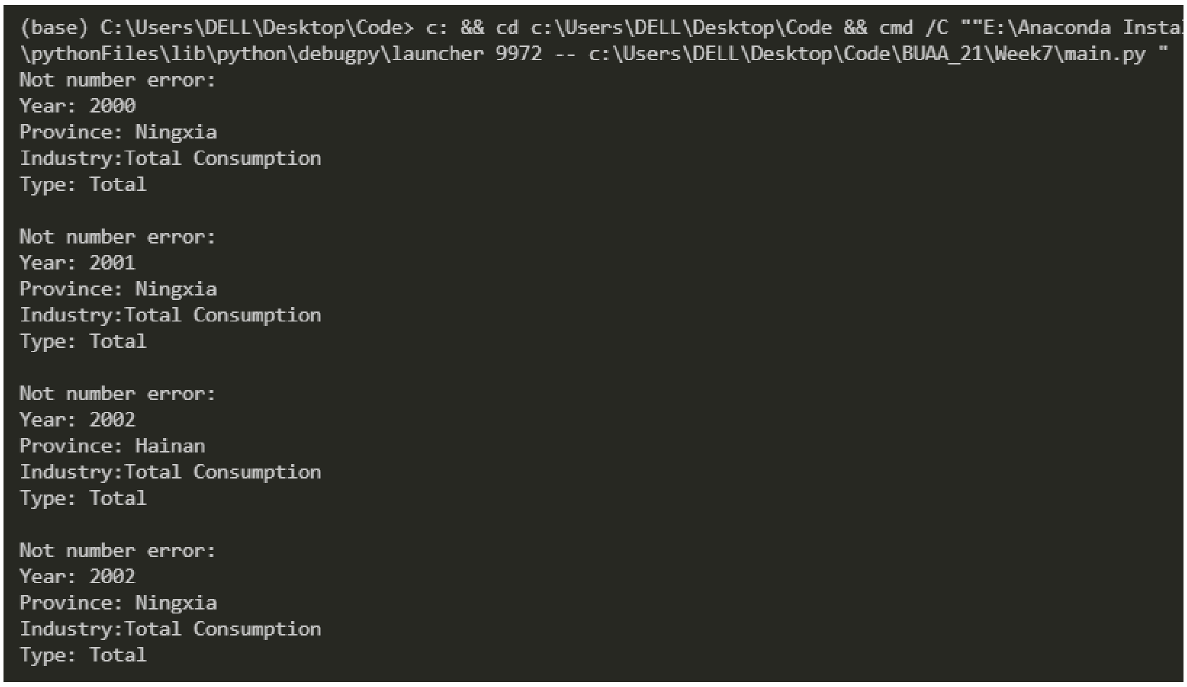

Raw Coal proportion of provinces in 1997
Jilin
Liaoning
InnerMongolia
Shanxi
Hebei
Tianjin
Beijing
Ningxia
Qinghai
Gansu
Shaanxi
Yunnan
Guizhou
Sichuan
Chongqing
Hainan
Guangxi
Guangdong
Hunan
Hubei
Henan
Shandong
Jiangxi
Fujian
Anhui
Zhejiang
Jiangsu
Shanghai
Heilongjiang



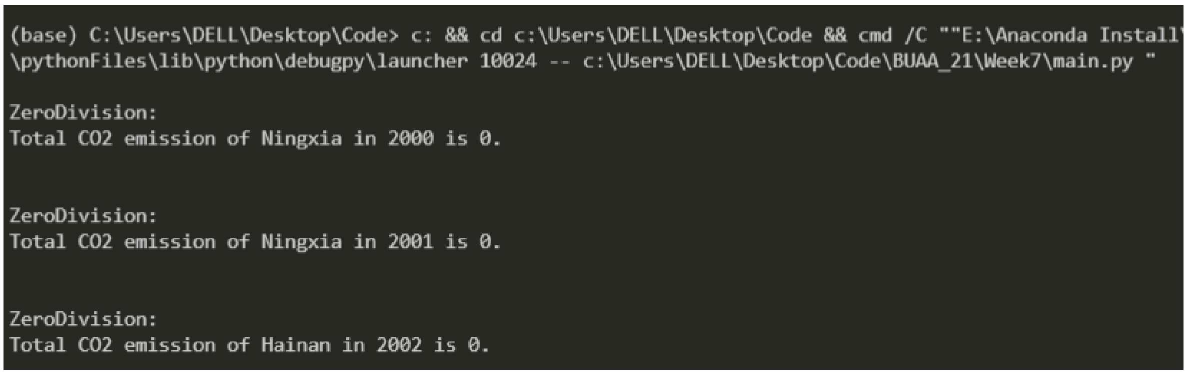


利用pyecharts生成的动态图片请查看文件夹中的html文件

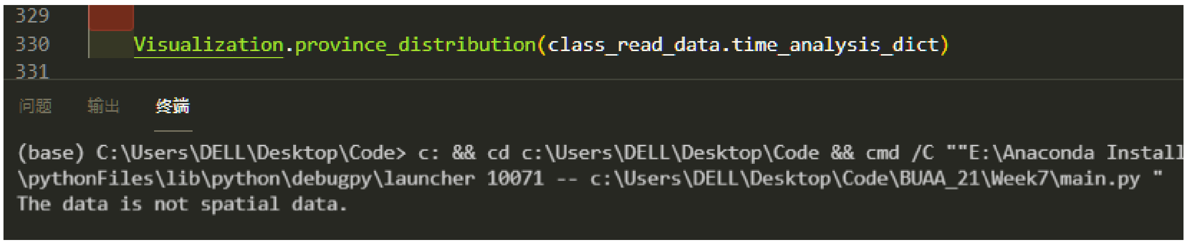
异常检测并打印错误信息



```
(base) C:\Users\DELL\Desktop\Code> c: && cd c:\Users\DELL\Desktop\Code && cmd /C ""E:\Anaconda Insta
\pythonFiles\lib\python\debugpy\launcher 9972 -- c:\Users\DELL\Desktop\Code\BUAA_21\Week7\main.py "
Not number error:
Year: 2000
Province: Ningxia
Industry:Total Consumption
Type: Total

Not number error:
Year: 2001
Province: Ningxia
Industry:Total Consumption
Type: Total

Not number error:
Year: 2002
Province: Hainan
Industry:Total Consumption
Type: Total

Not number error:
Year: 2002
Province: Ningxia
Industry:Total Consumption
Type: Total
```



```
(base) C:\Users\DELL\Desktop\Code> c: && cd c:\Users\DELL\Desktop\Code && cmd /C ""E:\Anaconda Install
\pythonFiles\lib\python\debugpy\launcher 10024 -- c:\Users\DELL\Desktop\Code\BUAA_21\Week7\main.py "

ZeroDivision:
Total CO2 emission of Ningxia in 2000 is 0.


ZeroDivision:
Total CO2 emission of Ningxia in 2001 is 0.


ZeroDivision:
Total CO2 emission of Hainan in 2002 is 0.
```



```
Visualization.province_distribution(class_read_data.time_analysis_dict)
```

```
(base) C:\Users\DELL\Desktop\Code> c: && cd c:\Users\DELL\Desktop\Code && cmd /C ""E:\Anaconda Install
\pythonFiles\lib\python\debugpy\launcher 10071 -- c:\Users\DELL\Desktop\Code\BUAA_21\Week7\main.py "
The data is not spatial data.
```

作业总结

通过本次作业，在数据可视化、异常处理等方面有了更深入的理解，同时也对pyecharts,matplotlib等库有了更多的了解，try-except-else-finally语句在程序中的使用，可以很好的定位程序发生错误的位置和错误类型，减少程序调试的时间，提高工作效率。

同时，在过往进行xlsx的处理大部分时间是用R的readxlsx包，在python上并不能使用，同时操作也比较麻烦。而在本次作业中，学习到了新的excel表格处理工具——xlrd库，对数据进行读取和清洗造作是较为方便的。

但是，在进行数据可视化的过程中，对数据的要求比较严苛，必须有指定形式的参数才能进行绘图等等，其实程序的可移植性是比较差的，对于这个数据集可能起到很好的作用，但是对于其他数据集效果则会大打折扣甚至直接报错。在今后的学习和作业中，我也会继续注意此方面的问题，养成良好的代码编写习惯。

参考资料

> https://www.cnblogs.com/auguse/articles/14108847.html 介绍xlrd库
> https://blog.csdn.net/weixin_44386638/article/details/85915993 利用matplotlib绘制折线图
> https://blog.csdn.net/mighty13/article/details/113898922 利用matplotlib绘制饼图
> https://blog.csdn.net/qq_39451578/article/details/104372597 pyecharts绘制热力图
> https://blog.csdn.net/qq_42346414/article/details/82149019 pyecharts timeline实例
> https://pyecharts.org/#/zh-cn/geography_charts pyecharts中文文档

相关文件

本次作业相关文件和代码已上传至北航云盘以及Github代码库