Statistics Final Project

Jianhuang Gan

2022 年 12 月 26 日

摘要

本文从两个方面探究北京 PM2.5 含量的影响因素，一是从 PM2.5 数据本身的周期，趋势等因素，以及自相关性，采用季节因素分解模型和 ARIMA 模型拟合。二是当期 PM2.5 含量也会受到当时其他变量的影响，采用线性回归模型拟合。为了综合考虑以上两方面及模型拟合效果，先采用季节因素分解，然后对残差拟合 ARIMAX 模型。最终发现 PM2.5 的显著影响因素有季节因素，露点，气压和温度。PM2.5 呈季节性，冬春季节高峰，夏秋低谷；越高的露点，PM2.5 含量越高，越高的气压和温度，PM2.5 的含量越低。

Contents

1 研究问题与数据处理

1.1 研究问题及其研究思路

本文尝试探究影响 PM2.5 的因素。本文认为影响 PM2.5 的因素主要来自两个方面：1.PM2.5 含量会受到季节，趋势等影响，采取季节因素分解模型探究，同时它作为时间序列数据，也会受到之前数据的影响，采用 ARIMA 对因素分解后的残差进行拟合。2.PM2.5 含量也会受到当时其他因素的影响，比如当时的气压，温度等因素的影响，采用线性回归模型进行拟合。最后，为了综合考虑以上两个方面，先进行季节因素分解，然后对残差采用 ARIMAX 模型进行拟合。

1.2 数据来源及插值处理

本文数据来源于 https://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data，北京每小时的气候观测数据，时间跨度从 2010.01.01 到 2014.12.31，总数据量 43824。pm2.5 数据缺失 2067 个，采取向前插值，用前一个数据值填充缺失值。但因为前 24 个数据为缺失值，向前填充无法填充数据，考虑到 24 个数据占比很小，丢弃前 24 个数据。插值处理后的数据为 2010.01.02.00 到 2014.12.31.23 共 43800 个数据。

数据总共变量有如下变量：将 year，month，day，hour 统一为 datetime 类型变量并作为 dataframe 的 index, 对于 cbwd，进行变量转换为 cbwd_cat，各取值为 0-3，分别对应 cbwd 的四种类型。下面表 1，表 2 分别展示处理前后的数据变量表。

表 1: 原始数据变量表

Orginal Variable	Descripition
No	row number
year	year of data in this row
month	month of data in this row
day	day of data in this row
hour	hour of data in this row
pm2.5	PM2.5 concentration ug/m^3
DEWP	Dew Point (â„f)
TEMP	Temperature (â„f)
PRES	Pressure (hPa)
cbwd	Combined wind direction
Iws	Cumulated wind speed (m/s)
Is	Cumulated hours of snow
Ir	Cumulated hours of rain

表 2: 处理后的数据变量表

Processed Variable	Descripition
Date index	year month day hour
pm2.5	PM2.5 concentration ug/m^3
DEWP	Dew Point (â„f)
TEMP	Temperature (â„f)
PRES	Pressure (hPa)
cbwd_cat	Wind direction category
Iws	Cumulated wind speed (m/s)
Is	Cumulated hours of snow
Ir	Cumulated hours of rain

1.3 数据描述性统计

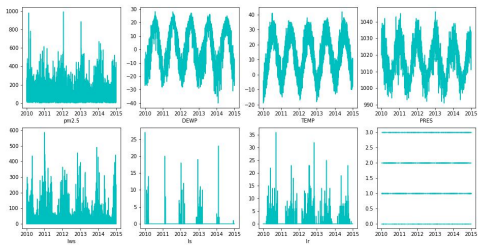
本部分展示处理后的部分数据（表 3），数据的整体性描述（表 4）以及各变量的时序分布图（a）和 PM2.5 的时序分布图（b）。

表 3: 处理后的部分数据展示

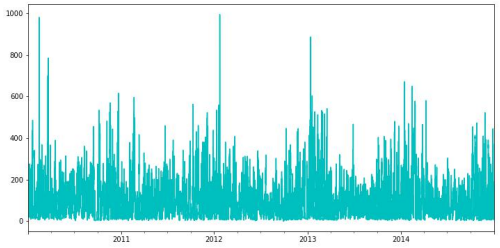
date_index	pm2.5	DEWP	TEMP	PRES	Iws	Is	Ir	cbwd_cat
2010-01-02 00:00:00	129.0	-16	-4.0	1020.0	1.79	0	0	2
2010-01-02 01:00:00	148.0	-15	-4.0	1020.0	2.68	0	0	2
2010-01-02 02:00:00	159.0	-11	-5.0	1021.0	3.57	0	0	2
2010-01-02 03:00:00	181.0	-7	-5.0	1022.0	5.36	1	0	2
2010-01-02 04:00:00	138.0	-7	-5.0	1022.0	6.25	2	0	2

表 4: 数据整体描述

	pm2.5	DEWP	TEMP	PRES	Iws	Is	Ir	cbwd_cat
count	43800.00	43800.00	43800.00	43800.00	43800.00	43800.00	43800.00	43800.00
mean	97.78	1.82	12.45	1016.44	23.89	0.05	0.19	1.66
std	91.39	14.42	12.19	10.27	50.02	0.76	1.41	0.93
min	0.00	-40.00	-19.00	991.00	0.45	0.00	0.00	0.00
25%	29.00	-10.00	2.00	1008.00	1.79	0.00	0.00	1.00
50%	72.00	2.00	14.00	1016.00	5.37	0.00	0.00	2.00
75%	136.00	15.00	23.00	1025.00	21.91	0.00	0.00	2.00
max	994.00	28.00	42.00	1046.00	585.60	27.00	36.00	3.00



(a) Variables distribution plot



(b) PM2.5

2 季节因素分解及 ARIMA

第一个方面：本文认为影响 PM2.5 的因素来自于自身，比如自相关性，自身呈现的周期性，季节性，趋势和随机扰动等因素。正如常理和图 (b)PM2.5 的时序图来看，PM2.5 分布情况或是峰值会因季节的不同而变得不同，也可能因为年份不同而存在不同的趋势可能，因此先采用季节因素分解模型。因为季节因素分解模型对于平稳和非平稳数据都能进行因素分解，本文先采取季节因素分解模型对数据进行因素分解，然后对因素分解后的残差进行白噪声检验，如果其为白噪声说明信息提取完毕，如果残差不为白噪声，说明残差中还存在信息未提取。一般而言，时间序列会受到临近数据的影响，存在自相关关系，所以本文对残差进行 ARIMA 模型拟合，进一步提取残差中的信息。

2.1 季节因素分解模型

由于数据是每小时数据，为了减少因为每日不同时刻的 PM2.5 波动问题，所以对数据进行 2*24 步的移动平均用于提取并消除时刻因素影响。从时序图上，可以发现数据的趋势性不明显，采取季节调整

的加法模型进行拟合。

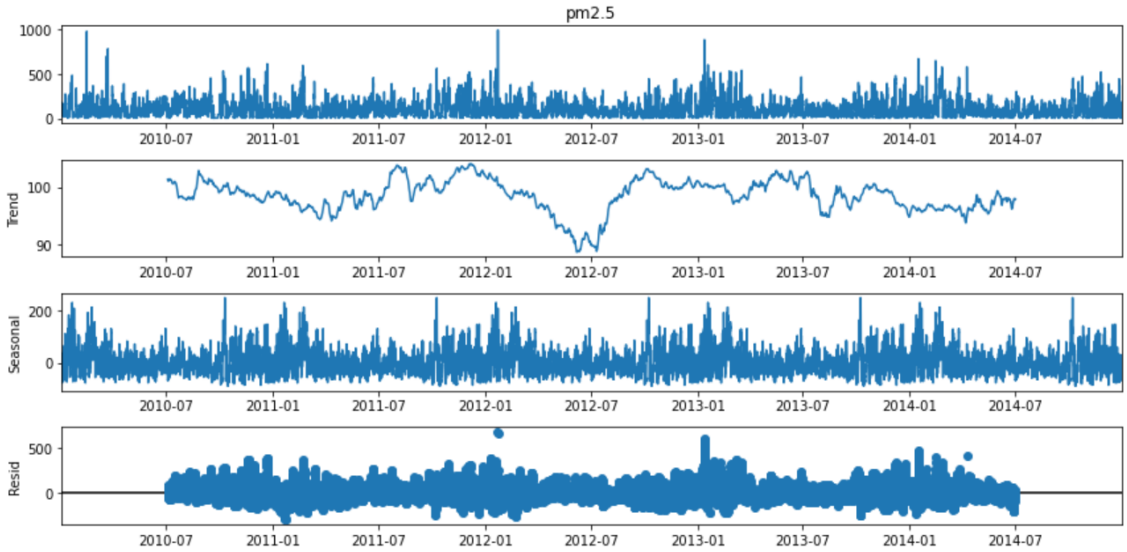


图 1: seasonal decompose

2.1.1 季节因素分解模型分析及残差白噪声检验

从图 1 中可以发现，季节因素被提取出来，对 PM2.5 的影响为 1 月高峰持续到 3 月，然后开始下降到 6 月，在 7，8 月份有小幅重新上涨，9，10 月略有下降，之后便一路上升到来年 1 月。可以认为，每年 PM2.5 在 11 月左右到来年 3 月左右为全年高峰期，其他月份有小幅波动，涨幅不明显，在 4 到 6 月左右和 8 到 10 月为低谷。大致体现为，PM2.5 在冬春两季达到高峰，而夏秋呈现低谷。猜测是可能因为冬春两季北京煤电供暖的原因导致。而所提取的趋势因素却没有显示出理论中的趋势现象，而是在不断的波动振荡，因此有理由怀疑是在不同时间段趋势有不同的表现，或是模型提取信息不足所导致的。不论哪种情况，季节因素分解模型的拟合情况都表现一般。

lag	lb_stat	lb_pvalue
1	40846.403209	0.0
2	77811.677784	0.0
3	111161.246925	0.0
4	141281.801319	0.0
5	168419.144415	0.0
6	192840.054872	0.0

表 5: 残差白噪声检验

对残差进行白噪声检验，采用 LB 滞后 6 期检验，因为一般情况下认为数据的自相关性是短期自相关，因此此处仅检验前 6 期。从表 5 的 LB 检验结果来看，6 个 P 值均小于 0.05，拒绝白噪声原假设，所以认为季节调整模型拟合效果较差，残差序列不为白噪声，信息提取不完全。

2.2 ARIMA 模型

2.2.1 平稳性检验及模型选择

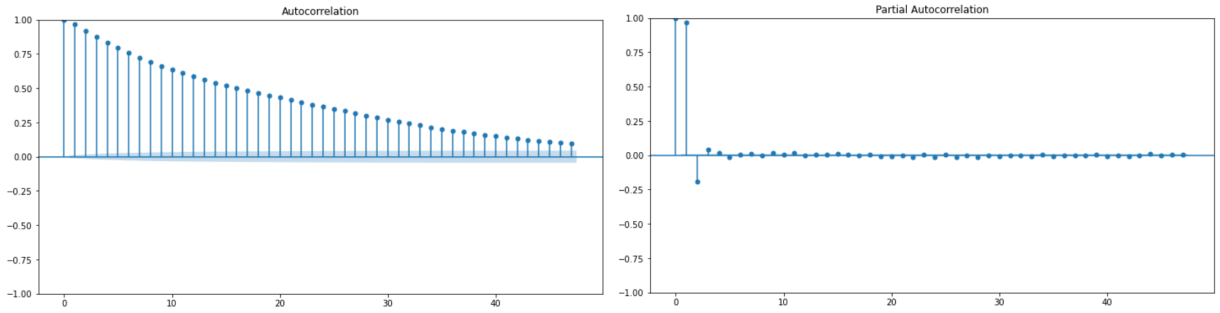


图 2: acf & pacf

对 PM2.5 进行 ADF 检验其平稳性：T=-21.21, p=0.0，从而拒绝原假设，认为序列平稳。同时，从 ACF 和 PACF 图中可以看出，ACF 不截断，而 PACF 也不截断，尝试拟合 ARMA(1-3,1-3)，并用 AIC,BIC 指标筛选最优模型。从表 6 中，可以看出 ARMA(1,3) 的 AIC 和 BIC 最小，选择拟合 ARIMA(1,0,3)。

表 6: ARIMA 模型选择

Model	AIC	BIC	Model	AIC	BIC	Model	AIC	BIC
ARMA(1,1)	392250.8	392285.6	ARMA(1,2)	392248.6	392292.0	ARMA(1,3)	392209.8	392261.9
ARMA(2,1)	392249.6	392293.0	ARMA(2,2)	392234.8	392286.9	ARMA(2,3)	392211.3	392272.1
ARMA(3,1)	392226.6	392278.8	ARMA(3,2)	392210.1	392270.9	ARMA(3,3)	392211.6	392281.1

2.2.2 ARIMA 模型拟合及部分预测效果

采用 ARMA(1，3) 进行模型拟合，拟合的 ARIMA(1,0,3) 模型如表 8 所示。然后对残差进行 LB 检验，如表 7 所示，从残差 LB 检验结果来看，残差已经基本上是白噪声序列，说明模型信息提取充分。同时因为数据量太多，所以仅将部分拟合预测效果展示在表 9 中，可以看出其拟合效果不错。

表 7: 残差白噪声检验 LB 滞后 6 期检验

lag	lb_stat	lb_pvalue
1	0.000083	0.992738
2	0.000083	0.999958
3	0.073823	0.994782
4	0.862386	0.929894
5	0.862676	0.972865
6	6.691459	0.350325

表 8: ARIMA(1,0,3)

var	coef	stderr	z	P
const	97.7840	2.988	32.728	0.000
ar.L1	0.9541	0.001	779.237	0.000
ma.L1	0.2006	0.002	103.405	0.000
ma.L2	0.0011	0.002	0.617	0.537
ma.L3	-0.0324	0.002	-16.062	0.000
AIC	392209.860	BIC	392261.985	

表 9: 部分拟合预测效果展示

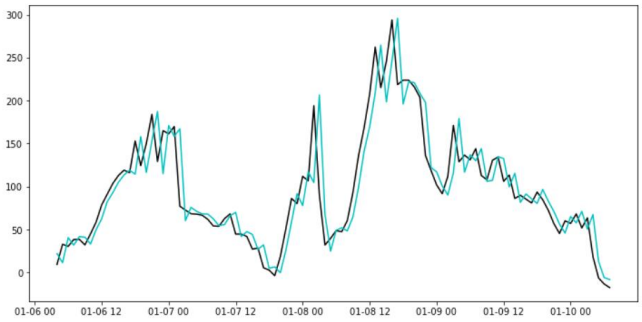


表 10: 多重共线性 VIF 检验

VIF	Factor	features
0	34655.111565	const
1	3.903416	DEWP
2	4.473352	TEMP
3	3.493725	PRES
4	1.152458	Iws
5	1.016077	Is
6	1.034782	Ir
7	1.085914	cbwd_cat

3 最小二乘线性回归

第二个方面，本文认为 PM2.5 会受到当期其他变量的影响，如天气，温度，气压等外生变量的影响而发生改变。在本部分，尝试采用线性回归模型去拟合模型，探究影响 PM2.5 的外生变量。

3.1 变量相关性矩阵图，多重共线性检验，协整关系检验

各变量的相关图如图 3 所示，并通过多重膨胀性因子 VIF 方法检验多重共线性，检验结果如表 10 所示，可知变量之间的 VIF 均小于 5，认为变量之间不存在多重共线性关系。为了避免伪回归问题，对数据进行协整关系检验，采用 EG 两步法 (EG two-stage), 对回归残差进行 ADF 检验，得到 t 值-187.044154，p 值 0.000，所以认为变量间存在协整关系，该回归不是伪回归。

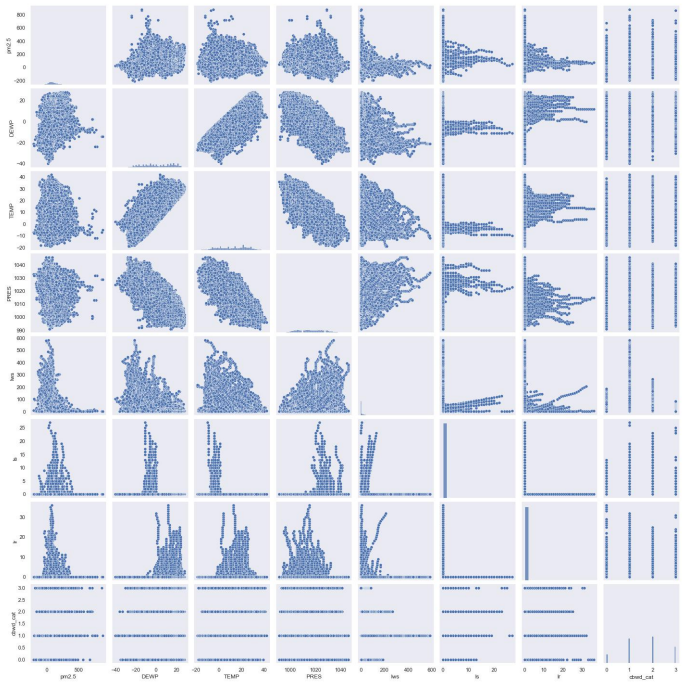


图 3: Variables Correlation

3.2 OLS 回归结果

从表 11 回归结果来看,各变量对于 PM2.5 的影响均为显著影响,其中正影响的变量有 DEWP,cbwd_cat，说明露点（空气中水汽达到饱和时的温度）上升时，空气中的湿度越高，PM2.5 的含量将越高；不同风向对于 PM2.5 也有不同的影响。剩余变量对 PM2.5 均为负影响，温度越高，气压越高，风速越快，雪的持续时间越久，雨的持续时间越久，PM2.5 的含量也将越小。这与生活常识相匹配，但回归拟合效果并不理想，Adjust R-square 仅为 0.238，说明 PM2.5 还有大量信息未被如上变量所蕴含。

表 11: OLS 回归结果

var	coef	stderr	t	P
const	1740.0427	70.956	24.523	0.000
DEWP	3.9531	0.052	75.744	0.000
TEMP	-5.8152	0.066	-87.954	0.000
PRES	-1.5653	0.069	-22.567	0.000
Iws	-0.2204	0.008	-26.941	0.000
Is	-2.2009	0.505	-4.357	0.000
Ir	-6.3301	0.274	-23.121	0.000
cbwd_cat	12.3860	0.423	29.251	0.000
Adj. R-squared	0.238	BIC	5.080e+05	

3.3 模型预测效果展示

将数据按 8：2 的比例分为训练集和测试集，用训练集训练模型，用测试集拟合预测结果，并选取部分拟合效果展示和残差的 QQ 图绘制于图 4，可以得出结果, 预测的拟合效果并不好，且残差也不遵从正态分布假设。所以该线性回归模型拟合效果不好。

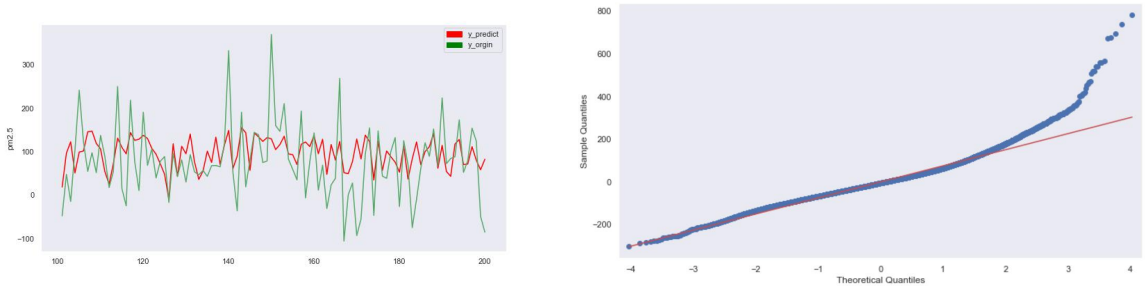


图 4: predict & Resid QQ plot

4 ARIMAX

由于季节分解 +ARIMA 模型只抓取了 PM2.5 本身的信息，而没有捕捉同期其他变量对于 PM2.5 的影响；相对的，线性回归模型捕捉了其他变量的影响，却没有捕捉变量自身相关性的影响。考虑以上，希望找到一种方法，既能同时捕获变量自身相关性因素和当期其他变量影响因素，所以最终考虑采用 ARIMAX 模型进行拟合，以同时捕获以上两个方面的信息，从而更好的捕捉 PM2.5 的信息。同样的，

考虑到小时因素和季节因素的影响，在拟合 ARIMAX 模型前，将对 PM2.5 数据进行 2*24 移动平均以消除小时信息影响，然后对其进行季节因素分解，最后对于分解后的数据进行 ARIMAX 模型拟合。

4.1 变量协整性检查

在进行 ARIMAX 模型拟合之前，变量需要满足协整性的要求，这里采用两种方法检验变量的协整性。一，在回归部分已经检验了协整关系，采用 EG 两步法 (EG two-stage), 对回归残差进行 ADF 检验，得到 t 值-187.044154，p 值 0.000，所以认为变量间存在协整关系。

二，若各变量均平稳则变量满足协整性，而不需要进行协整性检验。这里先对各变量进行平稳性进行 ADF 平稳性检验，得到表 12，可知每个变量均平稳，所以变量的协整性成立。

表 12: 各变量 ADF 检验

name	pm2.5	DEWP	TEMP	PRES	Iws	Is	Ir	cbwd_cat
Test Statis	-32.5	-6.2e+00	-3.9	-7.3e+00	-34.1	-46.7	-48.2	-50.0
p-value	0.0	5.2e-08	0.001771	7.9e-11	0.0	0.0	0.0	0.0
Lags Used	2.0	4.2e+01	49.0	5.5e+01	2.0	5.0	0.0	7.0
Number of Obs	43797.0	4.3e+04	43750.0	4.3e+04	43797.0	43794.0	43799.0	43792.0

4.2 ARIMAX 模型及结论

表 13: ARIMAX 模型选择

Model	AIC	BIC	Model	AIC	BIC
ARIMAX(0,0)	503053.6	503131.8	ARIMAX(0,1)	458030.1	458117.0
ARIMAX(0,2)	432289.9	432289.9	ARIMAX(0,3)	418542.4	418646.7
ARIMAX(1,0)	393325.2	393412.1	ARIMAX(1,1)	391938.9	392034.4
ARIMAX(1,2)	391941.8	392046.1	ARIMAX(1,3)	391886.0	391999.0
ARIMAX(2,0)	392006.3	392101.8	ARIMAX(2,1)	391944.6	392048.9
ARIMAX(2,2)	392430.3	3392543.2	ARIMAX(2,3)	391928.6	392050.2
ARIMAX(3,0)	391919.3	392023.5	ARIMAX(3,1)	393058.8	393171.7
ARIMAX(3,2)	393171.7	392044.8	ARIMAX(3,3)	391945.0	392075.4

从第二部分的图 2（ACF 和 PACF 图）中可以看出，PM2.5 的 ACF 和 PACF 均不截断，因此尝试拟合 ARIMAX(0-3,0,0-3)，采用 AIC,BIC 定阶，展示于表 13。ARIMAX(1,0,3) 的 AIC,BIC 最小，因此，选取 ARIMAX(1,0,3) 模型, 模型拟合如表 14 所示。残差 LB 检验如表 15 所示，说明残差已经是白噪声，模型信息提取完全。

var	coef	stderr	z	P
const	1603.8758	134.703	11.907	0.000
DEWP	1.0398	0.058	17.921	0.000
TEMP	-0.3469	0.064	-5.442	0.000
PRES	-1.4786	0.132	-11.187	0.000
Iws	-0.0149	0.007	-2.206	7 0.027
Is	-0.0459	0.492	-0.093	0.926
Ir	-0.2351	0.209	-1.123	0.262
cbwd_cat	-0.0157	0.090	-0.175	0.861
ar.L1	0.9554	0.001	785.792	0.000
ma.L1	0.1833	0.002	93.396	0.000
ma.L2	-0.0117	0.002	-6.508	0.000
ma.L3	-0.0394	0.002	-19.747	0.000
AIC	391886.074	BIC	391998.637	

表 14: ARIMAX(1,0,3)

lag	lb_stat	lb_pvalue
1	0.035392	0.850776
2	0.065627	0.967719
3	0.079288	0.994201
4	0.298845	0.989889
5	1.234748	0.941501
6	11.989339	0.062207

表 15: 残差白噪声检验 LB 滞后 6 期

从拟合的 ARIMAX 模型可以得到结论：对于 PM2.5 有显著正影响的变量是 DEWP，说明露点越高时，空气中的湿度越高，这也导致 PM2.5 的含量将越高, 显著负影响的变量有 TEMP,PRES, 越高的气温，越高的气压会有越低的 PM2.5 含量，这与线性回归部分的结论相同，不同的是其他变量在 ARIMAX 模型中不显著，而在 OLS 模型中显著。可以认为除了以上三个变量外，其他变量的影响都可由 PM2.5 自身，如自相关性等信息所解释。

5 结论及建议

综合以上三个模型，可以得到结论如下：（一）PM2.5 含量呈现周期性，大致表现为冬春两季高峰，夏秋低谷。（二）当期 PM2.5 含量受到之前 PM2.5 含量的影响，表现出自相关性。（三）PM2.5 含量还受到露点，气压和温度的影响。露点越高，PM2.5 含量越高；气压和温度越高，PM2.5 含量越低。（四）在 OLS 回归中显著的变量在 ARIMAX 模型中不显著的变量有：Iws,Is,Ir,cbwd_cat, 说明风速，积雪积雨量，风向的信息可被 PM2.5 自身或露点，气压，温度提取。（五）结论三同时也能解释结论一，一般来讲，冬春两季的露点低于夏秋，冬春两季的温度高于夏秋，冬春的气压也是高于夏秋的，部分体现为季节性影响。（六）在提取完季节因素影响后，露点，气压和温度对于 PM2.5 的影响依旧显著，说明这三个变量的影响不止表现为季节性周期，而其影响更为广泛和基础。

附录

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
import statsmodels.api as sm
from statsmodels.stats.diagnostic import acorr_ljungbox
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.model_selection import train_test_split
import matplotlib.patches as mpatches
from statsmodels.graphics.gofplots import qqplot

path = "D:/lecture/研一上学期/统计基础/Final_Project/PRSA.csv"
data = pd.read_csv(path)

# 将时间变量作为index
index = pd.date_range('2010-01-01', periods=43824, freq='H')
# del data['year']
data = data.set_index(index)

# 缺失值统计
print("缺失值统计: ", data.isnull().sum())
# 有缺失值, 用前值填充, 所以
data.fillna(method = 'ffill', inplace=True)
print(data.shape)
data = data.dropna()
print(data.shape)

# 将cbwd转为种类变量 0-3四种方向
data['cbwd_cat']=pd.Categorical(data.cbwd).codes

# 变量处理
# 删除year,month,day,hour（因为已经生成了对应的时间戳作为index
del data['year'], data['day'], data['hour'], data['month'], data['No'], data['cbwd']

# 描述统计部分
# 各变量类型
print(data.dtypes)
# 可视化各变量数据
fig = plt.figure(figsize=(16,8))
sublist = [241,242,243,244,245,246,247,248]
inx = 1
for i in data.columns:
    plt.subplot(sublist[inx-1])
    inx = inx+1
    if i == 'cbwd_cat':
        plt.scatter(data.index,data['cbwd_cat'] ,color='c', s=0.01)
        break

    plt.plot(data.index, data[i],color='c')
    plt.xlabel(i)

fig.savefig('D:/Latex_Project/Statistics_Project/Figures/var_plot.jpg')

# 为了能够进行季节性因素分解, 因为数据存在变量值为0的点, 将其赋值为0.00001以便分解
```

```
data = data.replace(0,0.00001)
data = data[:]

# 2*24的移动平均
rolling = data['pm2.5'].rolling(window=24)
rolling_mean = rolling.mean()

# 再用季节因素分解
result2=seasonal_decompose(data['pm2.5'], model='add', period=24*365)
plt.rcParams["figure.figsize"] = (12,6)
result2.plot()
fig.savefig('D:/Latex_Project/Statistics_Project/Figures/seasonal_decompose.jpg')
plt.show()

# 对残差进行白噪声检验
resid = result2.observed-result2.seasonal

print(acorr_ljungbox(resid, lags=6))
# 结果均为显著，说明不是残差序列不是白噪声序列，进一步对其进行ARIMA模型拟合，提取其自相关和累积相关关系

ARIMA 部分
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
#平稳性检测
from statsmodels.tsa.stattools import adfuller
plt.plot(resid)
# ADF检验平稳性
print('原始序列的检验结果为: ',adfuller(resid))
# 图看出大概也是平稳的,且ADF检验p值小于0.05，可以认为是平稳序列

# resid的ACF & PACF
fig = plt.figure(figsize=(1,6))
plot_acf(resid)
plot_pacf(resid)
# plt.show()
fig.savefig('D:/Latex_Project/Statistics_Project/Figures/acf_pacf.jpg')

# AIC,BIC定阶
from statsmodels.tsa.arima.model import ARIMA
# ARIMA(resid, order=(2,0,0)).fit()
pmax = 3
qmax = 3
aic_matrix = []
bic_matrix = []
for p in range(pmax):
    tempaic= []
    tempbic= []
    for q in range(qmax):
        try:
            tempaic.append(ARIMA(resid, order=(p+1, 0, q+1)).fit().aic)
            tempbic.append(ARIMA(resid, order=(p+1, 0, q+1)).fit().bic)
        except:
            tempaic.append(None)
            tempbic.append(None)
    aic_matrix.append(tempaic)
    bic_matrix.append(tempbic)
```

```
aic_table = pd.DataFrame(aic_matrix)   #将其转换成Dataframe 数据结构
p,q = aic_table.stack().idxmin()   #先使用stack 展平， 然后使用 idxmin 找出最小值的位置
print(u'AIC 最小的p值 和 q 值: \%s,\%s' \%(p+1,q+1))  #  AIC 最小的p值 和 q 值: 0,1
print(aic_table)

bic_table = pd.DataFrame(bic_matrix)   #将其转换成Dataframe 数据结构
p,q = bic_table.stack().idxmin()   #先使用stack 展平， 然后使用 idxmin 找出最小值的位置
print(u'BIC 最小的p值 和 q 值: \%s,\%s' \%(p+1,q+1))  #  BIC 最小的p值 和 q 值: 0,1
bic_table

# 拟合ARMA(1,3)模型
model = ARIMA(resid, order=(1, 0, 3))
model_fit = model.fit()

# 因为数据太多，截取部分数据观察拟合效果
predictions = model_fit.predict()
plt.plot(resid[100:200],color='black')
plt.plot(predictions[100:200], color='c')
# 但拟合效果并不怎么理想，有明显的滞后情况

# 对残差进行白噪声检验
print(acorr_ljungbox(model_fit.resid, lags=6))
# 此时残差已经几乎是白噪声了
model_fit.summary()

回归部分
# 变量相关性检验
import seaborn as sns

sns.set(style='dark', context='notebook')
sns.pairplot(data)
plt.savefig("D:/Latex_Project/Statistics_Project/Figures/var_corr.jpg")
plt.show()

# 部分变量存在线性关系，检查多重共线性问题
# 多重共线性检验
# 方差膨胀因子检验多重共线性
from statsmodels.stats.outliers_influence import variance_inflation_factor
import pandas as pd
def vif(data):
    features = ['DEWP','TEMP','PRES','Iws','Is','Ir','cbwd_cat']
    labels = ['pm2.5']
    Y = data[labels]
#     加入常量变量
    X = sm.add_constant(data[features])
    vif = pd.DataFrame()
    vif["VIF Factor"]=[variance_inflation_factor(X.values,i) for i in range(X.shape[1])]
    vif["features"]=X.columns
    print(vif)

vif(data)
# vif在5以下，认为没有多重共线性，可以进行OLS模型拟合
# 分析线性回归
import statsmodels.api as sm
```

```
def linearModel(data):
    features = ['DEWP','TEMP','PRES','Iws','Is','Ir','cbwd_cat']
    labels = ['pm2.5']
    Y = data[labels]
#     加入常量变量
    X = sm.add_constant(data[features])
#     构建模型
    re = trainModel(X, Y)
#     分析模型效果
    modelSummary(re)

def trainModel(X, Y):
    model = sm.OLS(Y, X)
    re = model.fit()
    return re

def modelSummary(re):
    print("整体统计分析结果",re.summary())

    coef_df = pd.DataFrame({"params": re.params,   # 回归系数
                        "std err": re.bse,     # 回归系数标准差
                        "t": round(re.tvalues,3),       # 回归系数T值
                        "p-values": round(re.pvalues,3) # 回归系数P值
                         })

    coef_df[['coef_0.025','coef_0.975']] = re.conf_int() # 回归系数置信区间 默认5%, 括号中可填具体数字
                                                          比如0.05, 0.1
    print(coef_df)

trainnum = int(data.shape[0]*0.80)
df_train = data.iloc[0:trainnum,:]
df_test = data.iloc[trainnum:,:]
linearModel(df_train)
# 从模型拟合结果来看, 虽然变量系数均显著, 但R^2较小, 拟合效果可能一般, 接下来拟合预测效果

def plot_predict(data,pred_range):
    features = ['DEWP','TEMP','PRES','Iws','Is','Ir','cbwd_cat']
    labels = ['pm2.5']
    Y = data[labels]
#     加入常量变量
    X = sm.add_constant(data[features])
#     构建模型
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size=0.8, random_state=123)

    re = trainModel(X_train, Y_train)

    Y_predict = re.predict(X_test)
    index = list(np.linspace(1,len(Y_predict),len(Y_predict)))
    color=['red','green']
    labels = ['y_predict', 'y_orgin']  #legend标签列表
    #用label和color列表生成mpatches.Patch对象, 它将作为句柄来生成legend
    patches = [mpatches.Patch(color=color[i], label="{:s}".format(labels[i]) ) for i in range(len(color
                                                          )) ]
    plt.plot(index[100:200],Y_predict[100:200],color='red')
```

```
#     plt.legend("Y_predict")
    plt.plot(index[100:200], Y_test['pm2.5'][100:200], color='g')
    plt.legend(handles=patches)
    plt.ylabel("pm2.5")
    plt.savefig("D:/Latex_Project/Statistics_Project/Figures/predict.jpg")
    plt.show()
#     残差QQ图检验正态性
    qqplot(re.resid, line='s')
    plt.savefig("D:/Latex_Project/Statistics_Project/Figures/qq.jpg")

#     协整检验，这里和后面ARIMAX要用
    ad = sm.tsa.adfuller(re.resid,autolag='BIC')
    ad_output = pd.Series(ad[0:4], index=['Test Statistic','p-value','Lags Used','Number of
                                                    Observations Used'])
    print(ad_output)
pred_range = [100,200]
plot_predict(data, pred_range)
# 拟合效果也一般。
# QQ图也不太像正态

ARIMAX部分
# 协整检验
# 直接检查各变量平稳性
# 因为ARIMAX(p,i,q)要求所有的序列的是平稳的，所以要对序列进行单位根检验，判断序列的平稳性。
## 1:单位根检验检验序列的平稳性,ADF 检验
data_pm25 = sm.tsa.adfuller(data['pm2.5'],autolag='BIC')
data_pm25_output = pd.Series(data_pm25[0:4], index=['Test Statistic','p-value','Lags Used','Number of
                                                Observations Used'])

data_DEWP = sm.tsa.adfuller(data['DEWP'],autolag='BIC')
data_DEWP_output = pd.Series(data_DEWP[0:4], index=['Test Statistic','p-value','Lags Used','Number of
                                                Observations Used'])

data_TEMP = sm.tsa.adfuller(data['TEMP'],autolag='BIC')
data_TEMP_output = pd.Series(data_TEMP[0:4], index=['Test Statistic','p-value','Lags Used','Number of
                                                Observations Used'])

data_PRES = sm.tsa.adfuller(data['PRES'],autolag='BIC')
data_PRES_output = pd.Series(data_PRES[0:4], index=['Test Statistic','p-value','Lags Used','Number of
                                                Observations Used'])

data_Iws = sm.tsa.adfuller(data['Iws'],autolag='BIC')
data_Iws_output=pd.Series(data_Iws[0:4],index=['Test Statistic','p-value','Lags Used','Number of
                                                Observations Used'])

data_Is = sm.tsa.adfuller(data['Is'],autolag='BIC')
data_Is_output = pd.Series(data_Is[0:4], index=['Test Statistic','p-value','Lags Used','Number of
                                                Observations Used'])

data_Ir = sm.tsa.adfuller(data['Ir'],autolag='BIC')
data_Ir_output = pd.Series(data_Ir[0:4], index=['Test Statistic','p-value','Lags Used','Number of
                                                Observations Used'])

data_cbwd_cat = sm.tsa.adfuller(data['cbwd_cat'],autolag='BIC')
data_cbwd_cat_output = pd.Series(data_cbwd_cat[0:4], index=['Test Statistic','p-value','Lags Used','
```

```
                                                Number of Observations Used'])

dic = {'pm2.5':data_pm25_output, 'DEWP':data_DEWP_output,'TEMP':data_TEMP_output,'PRES':
                                                data_PRES_output,'Iws':data_Iws_output,
       'Is':data_Is_output,'Ir':data_Ir_output,'cbwd_cat':data_cbwd_cat_output}

dic=pd.DataFrame(dic)
# 模型预设
result2=seasonal_decompose(data['pm2.5'], model='add', period=24*365)
plt.rcParams["figure.figsize"] = (12,6)
result2.plot()
# fig.savefig('D:/Latex_Project/Statistics_Project/Figures/seasonal_decompose.jpg')
plt.show()
data['pm2.5'] = result2.observed-result2.seasonal  #重新赋值
endog = data['pm2.5'][:]
exog = sm.add_constant(data[['DEWP','TEMP','PRES','Iws','Is','Ir','cbwd_cat']][:])

# AIC,BIC定阶
# df
pmax = 4
qmax = 4
aic_matrix = []
bic_matrix = []
for p in range(pmax):
    tempaic= []
    tempbic= []
    for q in range(qmax):
        try:
            tempaic.append(sm.tsa.statespace.SARIMAX(endog, exog, order=(p,0,q)).fit().aic)
            tempbic.append(sm.tsa.statespace.SARIMAX(endog, exog, order=(p,0,q)).fit().bic)
        except:
            tempaic.append(None)
            tempbic.append(None)
    aic_matrix.append(tempaic)
    bic_matrix.append(tempbic)

aic_table = pd.DataFrame(aic_matrix)   #将其转换成Dataframe 数据结构
p,q = aic_table.stack().idxmin()   #先使用stack 展平， 然后使用 idxmin 找出最小值的位置
print(u'AIC 最小的p值 和 q 值: \%s,\%s' \%(p,q))  #  AIC 最小的p值 和 q 值: 0,1
print(aic_table)

bic_table = pd.DataFrame(bic_matrix)   #将其转换成Dataframe 数据结构
p,q = bic_table.stack().idxmin()   #先使用stack 展平， 然后使用 idxmin 找出最小值的位置
print(u'BIC 最小的p值 和 q 值: \%s,\%s' \%(p,q))  #  BIC 最小的p值 和 q 值: 0,1
bic_table

# 定（1, 0, 3）
# Fit the model
mod = sm.tsa.statespace.SARIMAX(endog, exog, order=(1, 0, 3))
res = mod.fit(disp=False)
print(res.summary())

# 残差LB检验
print(acorr_ljungbox(res.resid, lags=6))
```