数据清洗

In [103…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

sns.set(style = "darkgrid" , font_scale = 1.2)
plt.rcParams["font.family"] = "SimHei"
plt.rcParams["axes.unicode_minus"] = "False"
plt.rcParams["font.size"] = 12  # 设置字体大小
warnings.filterwarnings("ignore")

# 设置 Seaborn 的样式和字体
sns.set(style="darkgrid", font="SimHei", rc={"axes.unicode_minus": False})
```

In [141…
```
data = pd.read_csv("data_228.csv", encoding="gbk")
print(data.shape)
data.head()
```

```
(325, 12)
```

Out[141…

	City	AQI	Precipitation	GDP	Temperature	Longitude	Latitude	Alt
0	Ngawa Prefecture	23	665.1	271.13	8.200000	102.224650	31.899410	2
1	Aksu City	137	80.4	610.00	12.276712	80.263380	41.167540	1
2	Alxa League	85	150.0	322.58	24.200000	105.728950	38.851920	1
3	Ngari	28	74.2	37.40	1.000000	80.105800	32.501110	4
4	Anqin City	79	2127.8	1613.20	17.291781	117.034431	30.512646	

In [67]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 325 entries, 0 to 324
Data columns (total 12 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   City                     325 non-null    object
 1   AQI                      325 non-null    int64
 2   Precipitation            321 non-null    float64
 3   GDP                      325 non-null    float64
 4   Temperature              325 non-null    float64
 5   Longitude                325 non-null    float64
 6   Latitude                 325 non-null    float64
 7   Altitude                 325 non-null    float64
 8   PopulationDensity        325 non-null    int64
 9   Coastal                  325 non-null    object
 10  GreenCoverageRate        325 non-null    float64
 11  Incineration(10,000ton)  0 non-null      float64
dtypes: float64(8), int64(2), object(2)
memory usage: 30.6+ KB
```

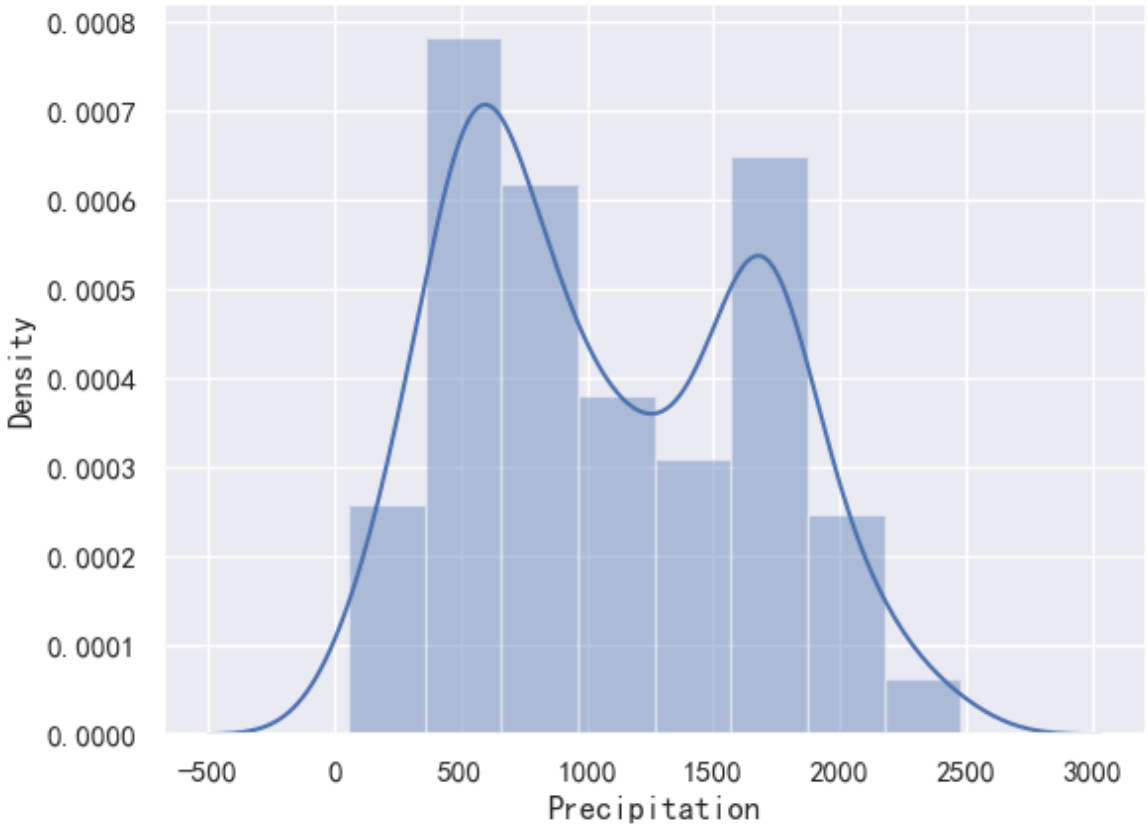
In [69]:
```
t = data.isnull().sum()
t = pd.concat([t,t/len(data)] , axis = 1)
t.columns = ["缺失值数量" , "缺失值比例"]
display(t)
```

	缺失值数量	缺失值比例
City	0	0.000000
AQI	0	0.000000
Precipitation	4	0.012308
GDP	0	0.000000
Temperature	0	0.000000
Longitude	0	0.000000
Latitude	0	0.000000
Altitude	0	0.000000
PopulationDensity	0	0.000000
Coastal	0	0.000000
GreenCoverageRate	0	0.000000
Incineration(10,000ton)	325	1.000000

In [71]:
```
print(data["Precipitation"].skew())
sns.distplot(data["Precipitation"].dropna())
```

```
0.27360760671177387
```

Out[71]: <Axes: xlabel='Precipitation', ylabel='Density'>



In [73]:
```
data.fillna({"Precipitation" : data["Precipitation"].median} , inplace = True)
data.isnull().sum()
```

Out[73]:
```
City                        0
AQI                         0
Precipitation               0
GDP                         0
Temperature                 0
Longitude                   0
Latitude                    0
Altitude                    0
PopulationDensity           0
Coastal                     0
GreenCoverageRate           0
Incineration(10,000ton)   325
dtype: int64
```

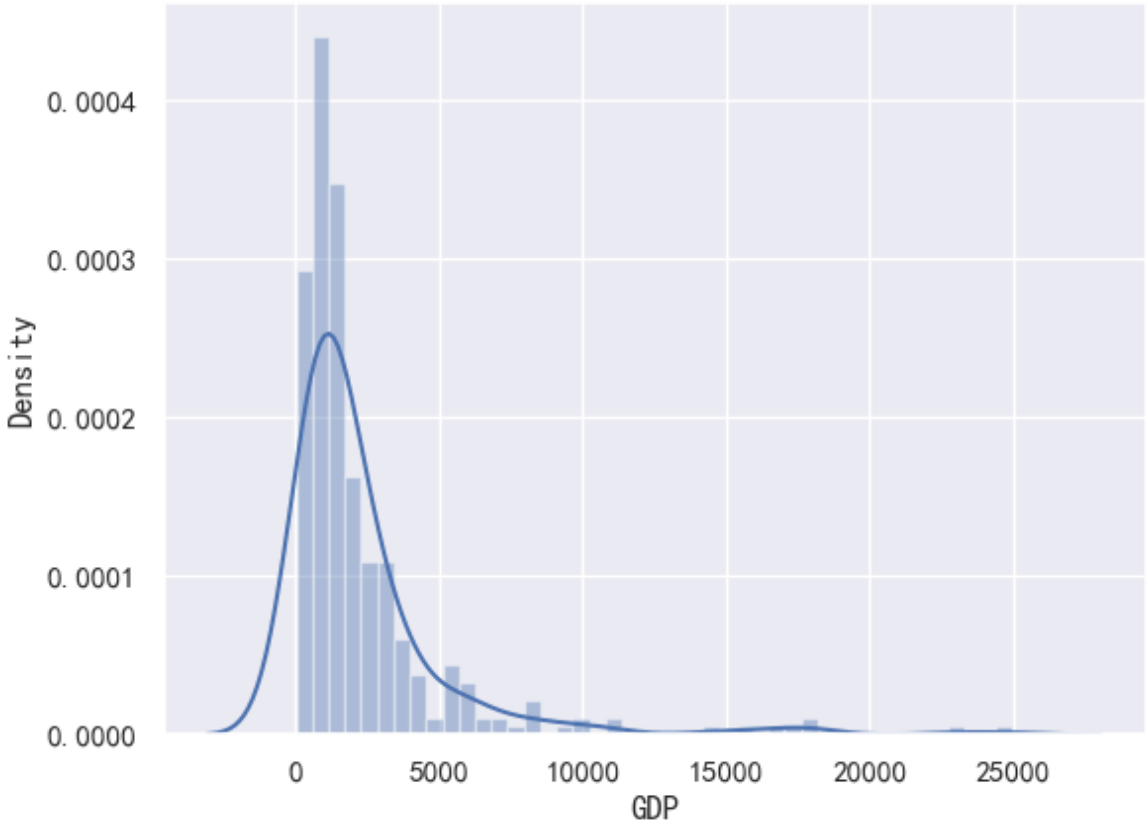
In [75]:
```
data.describe()
```

Out[75]:

	AQI	GDP	Temperature	Longitude	Latitude	Altituc
count	325.000000	325.000000	325.000000	325.000000	325.000000	325.00000
mean	75.809231	2390.901815	15.980149	113.990609	31.870665	380.14184
std	43.610516	3254.876921	5.016133	7.688515	6.093703	741.40970
min	12.000000	22.500000	-2.500000	80.105800	18.234043	-12.00000
25%	45.000000	762.970000	13.727397	111.130000	27.695387	18.00000
50%	69.000000	1328.520000	16.494521	115.500183	31.385597	62.00000
75%	102.000000	2735.340000	18.921918	119.823308	36.449432	354.00000
max	296.000000	24964.990000	27.447945	129.598496	49.220000	4505.00000

In [77]:
```
sns.distplot(data["GDP"])
print(data["GDP"].skew())
```

```
3.7614282419643033
```



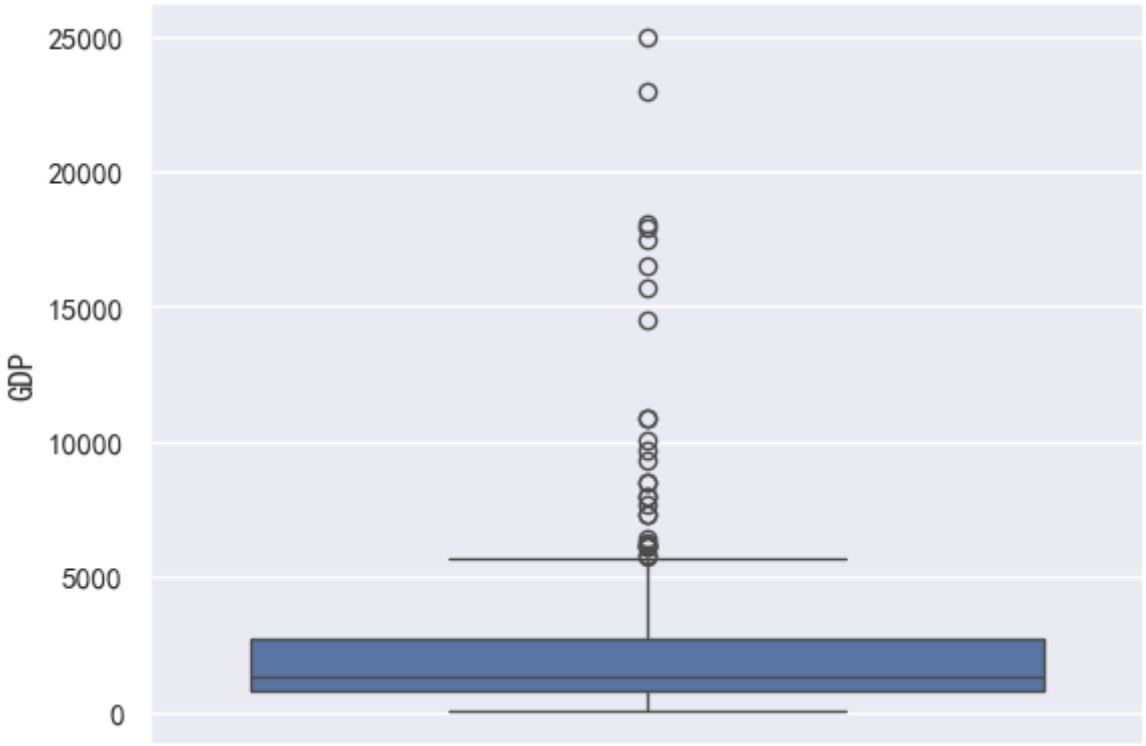
In [79]:
```
mean , std = data["GDP"].mean() , data["GDP"].std()
lower , upper = mean - 3*std , mean + 3*std
print(" 均值: " , mean)
print("标准差: " , std)
print(" 下限: " , lower)
print(" 上限: " , upper)

#获取在3倍标准差之外的数据
data["GDP"][(data["GDP"] < lower) | (data["GDP"] > upper)]
```

```
均值：  2390.9018153846155
标准差：  3254.876921271434
下限：  -7373.728948429687
上限：  12155.532579198918
```

Out[79]:
```
16     22968.60
63     18100.41
202    24964.99
207    17502.99
215    14504.07
230    16538.19
256    17900.00
314    15719.72
Name: GDP, dtype: float64
```

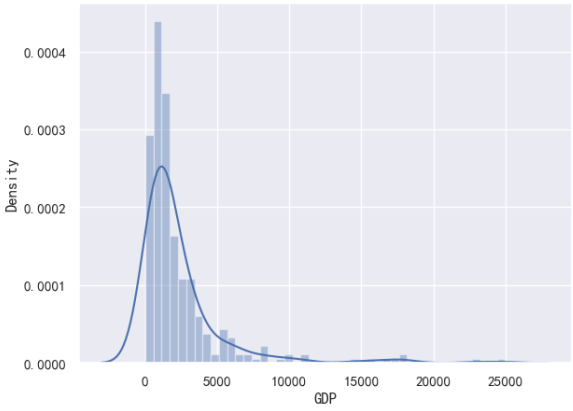
In [81]:
```
# 异常检测
sns.boxplot(data = data["GDP"])
```

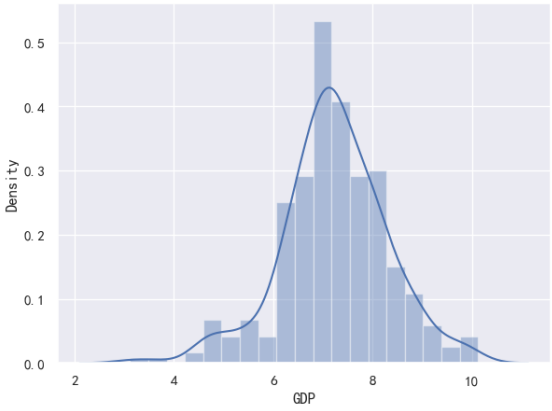
Out[81]:
```
<Axes: ylabel='GDP'>
```



In [83]:
```
fig , ax = plt.subplots(1 , 2)
fig.set_size_inches(15 , 5)
sns.distplot(data["GDP"], ax = ax[0])
sns.distplot(np.log(data["GDP"]), ax = ax[1])
```

Out[83]:
```
<Axes: xlabel='GDP', ylabel='Density'>
```





In [85]:
```
# 重复值
print(data.duplicated().sum())

data[data.duplicated( keep = False)]
```

2

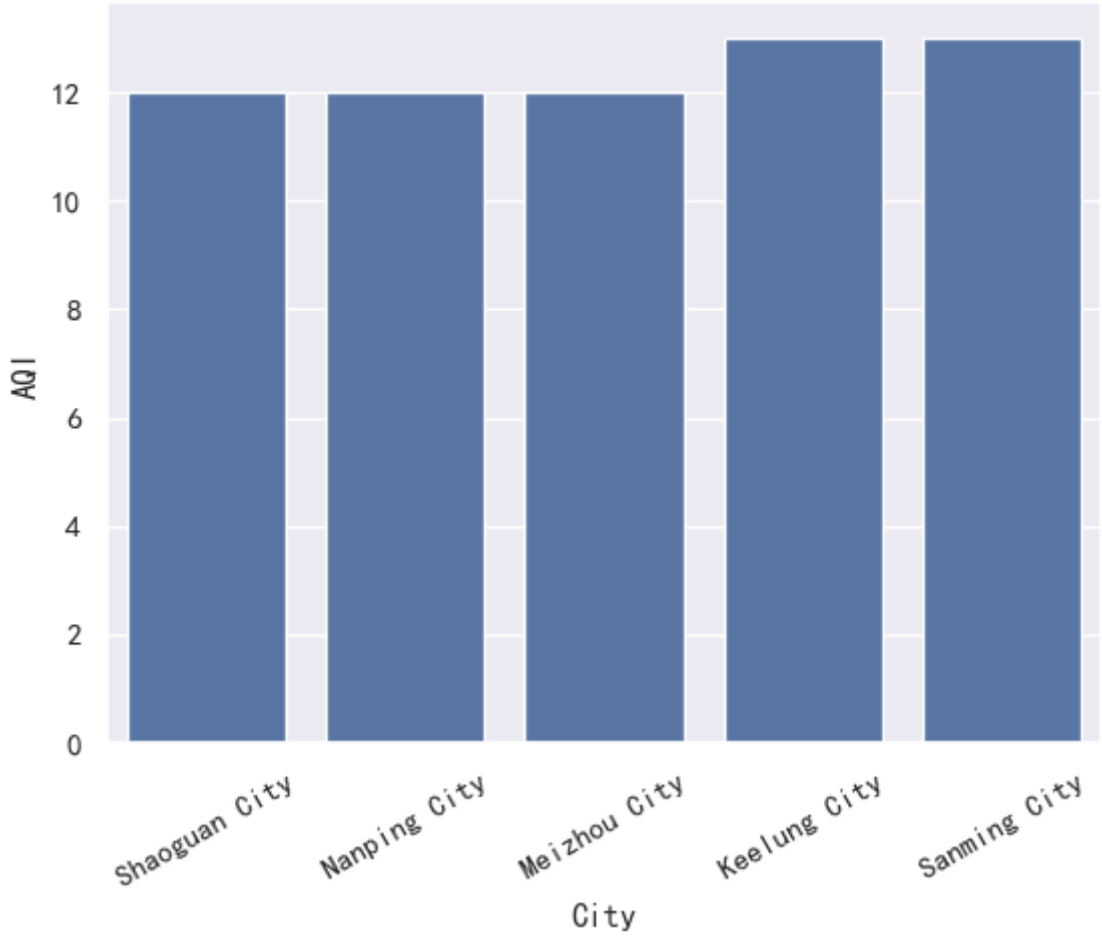
Out[85]:

	City	AQI	Precipitation	GDP	Temperature	Longitude	Latitude	AI
13	Baoding City	220	566.9	2757.80	13.258904	115.500183	38.857071	
109	Baoding City	220	566.9	2757.80	13.258904	115.500183	38.857071	
149	Luohe City	85	831.0	992.85	15.704110	114.041092	33.572510	
218	Luohe City	85	831.0	992.85	15.704110	114.041092	33.572510	

In [87]:
```
# 空气质量较好/较差的城市
t = data[["City" , "AQI"]].sort_values("AQI")
t = t.iloc[:5]
display(t)
plt.xticks(rotation = 30)
sns.barplot(x = "City" , y = "AQI", data = t)
```

	City	AQI
204	Shaoguan City	12
163	Nanping City	12
154	Meizhou City	12
91	Keelung City	13
195	Sanming City	13

Out[87]: <Axes: xlabel='City', ylabel='AQI'>

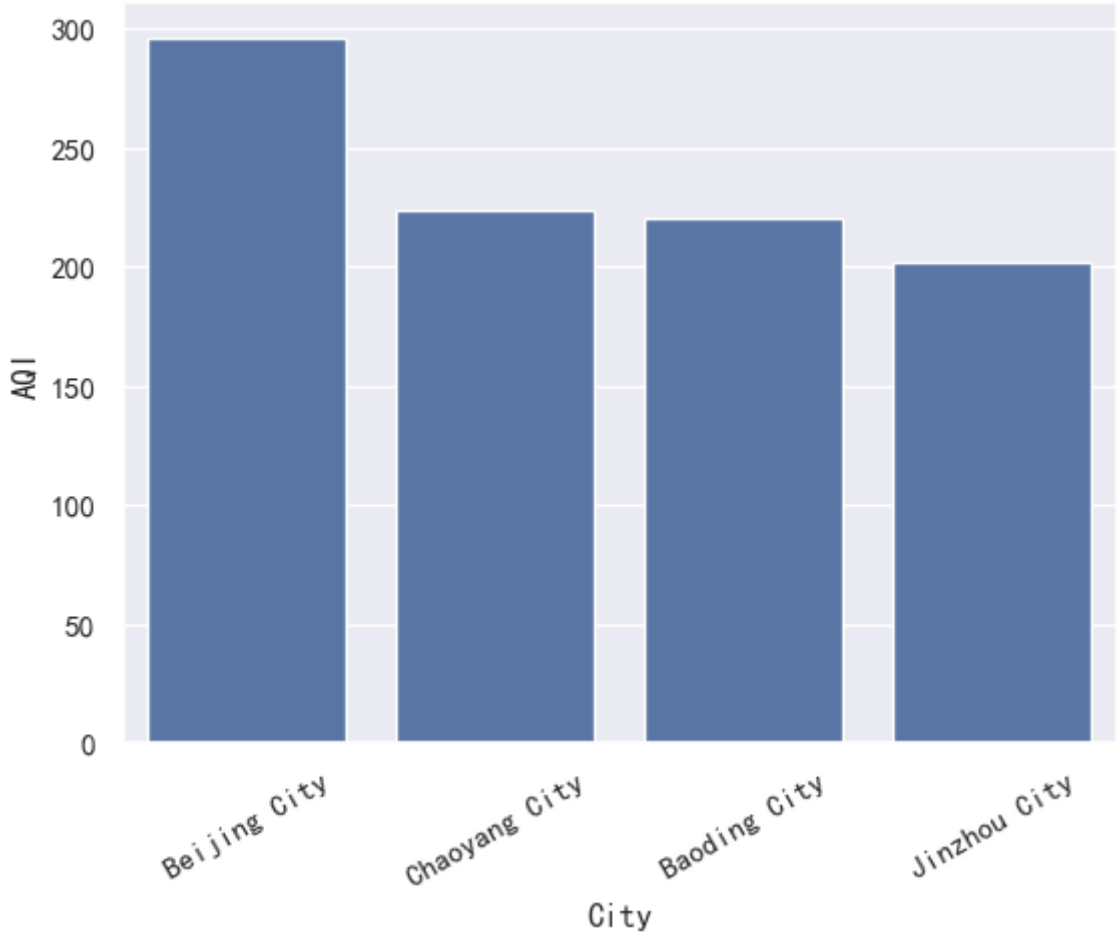


空气质量较好/较差的城市

In [90]:
```
t = data[["City" , "AQI"]].sort_values("AQI" , ascending = False)
t = t.iloc[:5]

display(t)
plt.xticks(rotation = 30)
sns.barplot(x = "City" , y = "AQI", data = t)
```

	City	AQI
16	Beijing City	296
26	Chaoyang City	224
13	Baoding City	220
109	Baoding City	220
112	Jinzhou City	202

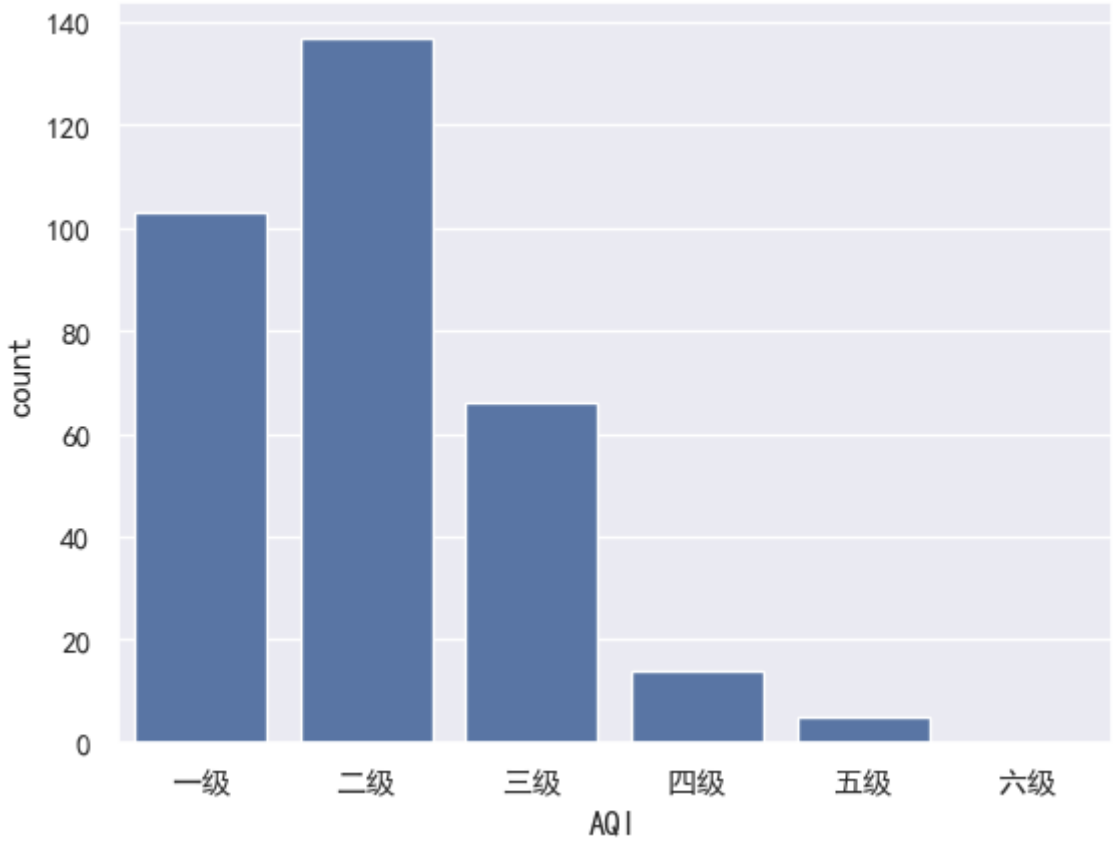
Out[90]: <Axes: xlabel='City', ylabel='AQI'>



In [92]:
```
# 全国空气质量等级
def value_to_level(AQI):
    if AQI >= 0 and AQI <=50:
        return "一级"
    elif AQI >= 51 and AQI <= 100:
         return "二级"
    elif AQI >= 101 and AQI <= 150:
         return "三级"
    elif AQI >= 151 and AQI <= 200:
         return "四级"
    elif AQI >= 201 and AQI <= 300:
         return "五级"
    else :
         return "六级"
level = data["AQI"].apply(value_to_level)
print(level.value_counts())
sns.countplot(x = level ,order = ["一级" , "二级", "三级", "四级", "五级", "六级
```

```
AQI
二级    137
一级    103
三级     66
四级     14
五级      5
Name: count, dtype: int64
```

Out[92]: <Axes: xlabel='AQI', ylabel='count'>



In [94]:
```
# 空气质量指数分布
sns.scatterplot(x = "Longitude" , y = "Latitude" , hue = "AQI" ,
                palette = plt.cm.RdYlGn_r , data = data)
```

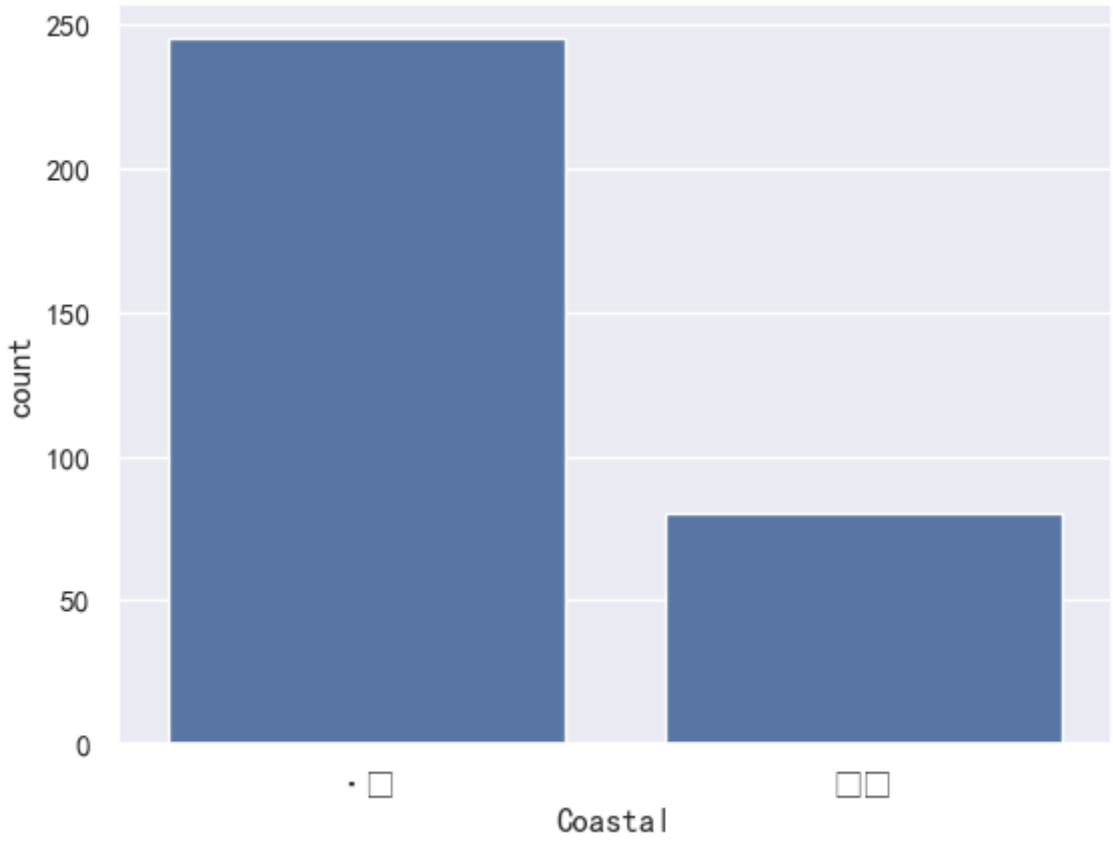
Out[94]: <Axes: xlabel='Longitude', ylabel='Latitude'>

临海城市空气质量是否优于内陆城市

In [113…
```
print(data["Coastal"].value_counts())
sns.countplot(x = "Coastal" , data = data)
```

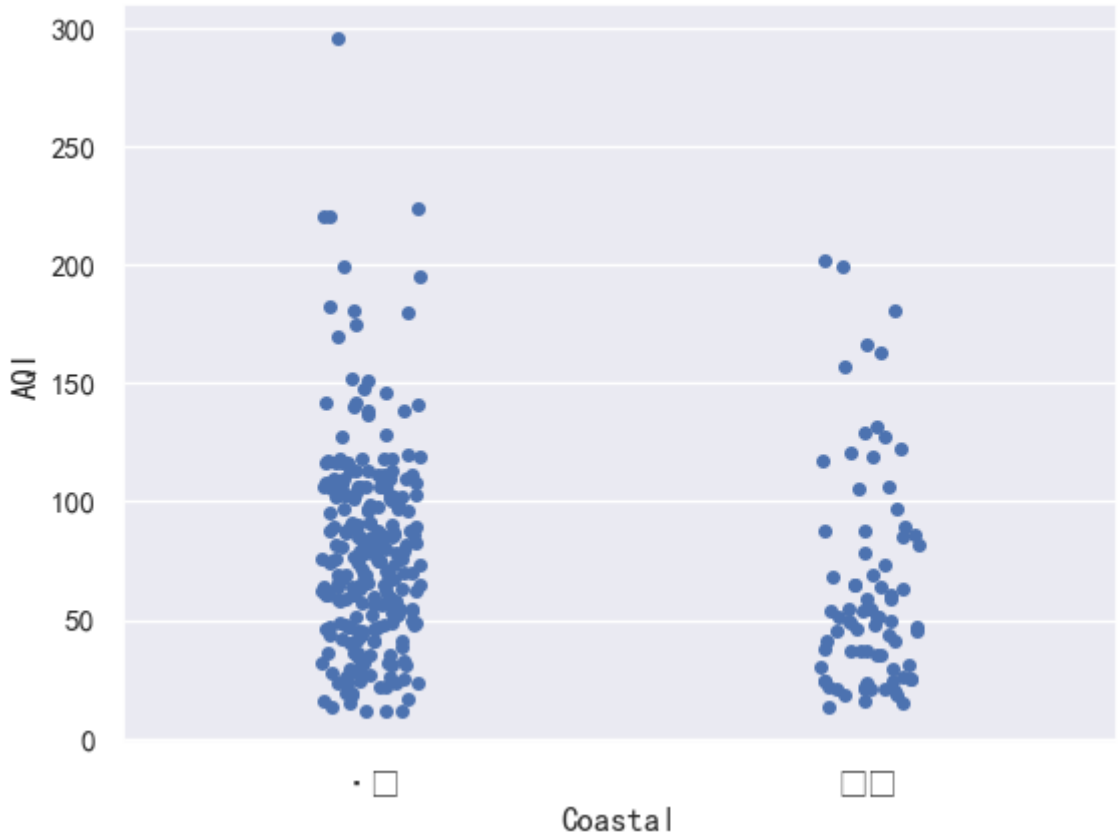
```
Coastal
·ñ    245
ÊÇ     80
Name: count, dtype: int64
```

Out[113… <Axes: xlabel='Coastal', ylabel='count'>



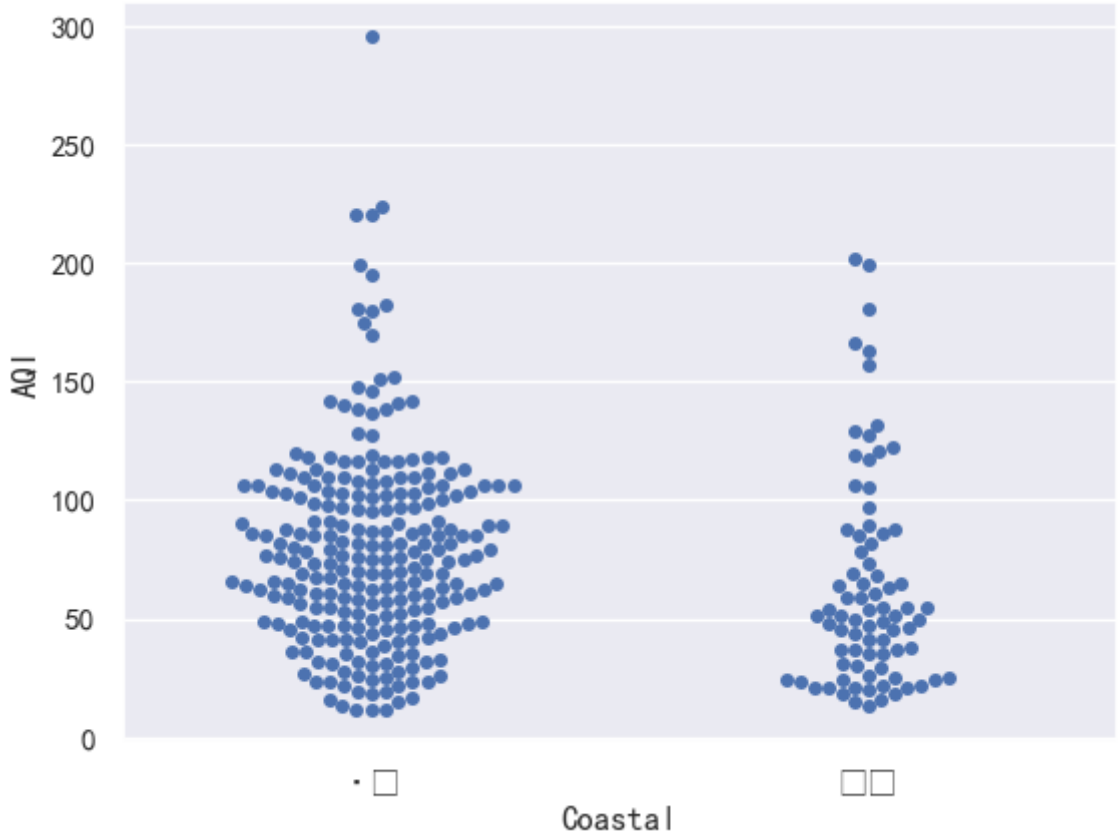
In [115…
```
sns.stripplot(x = "Coastal", y = "AQI", data = data)
```

Out[115… <Axes: xlabel='Coastal', ylabel='AQI'>



In [117…
```
sns.swarmplot(x = "Coastal", y = "AQI", data = data)
```

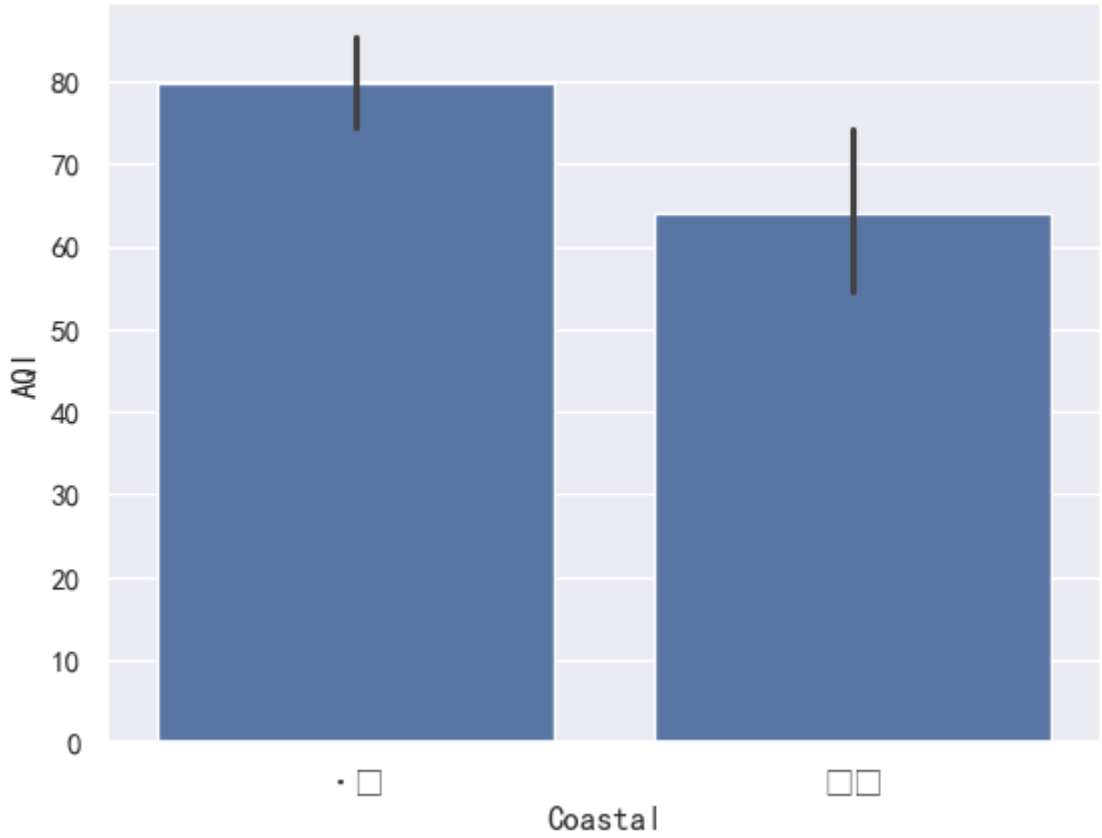
Out[117…
```
<Axes: xlabel='Coastal', ylabel='AQI'>
```



In [119…
```
print(data.groupby("Coastal")["AQI"].mean())
sns.barplot(x = "Coastal" , y = "AQI" , data = data)
```

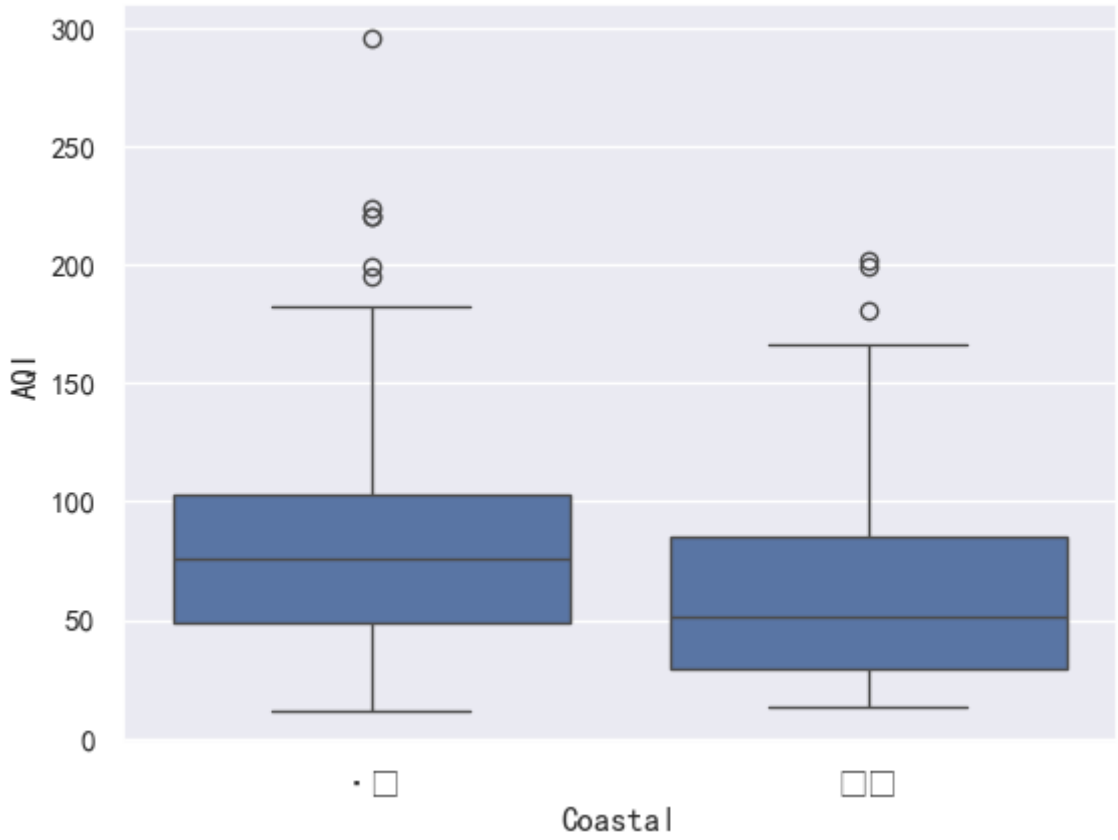
```
Coastal
·ñ    79.644898
ÊÇ    64.062500
Name: AQI, dtype: float64
```

Out[119… <Axes: xlabel='Coastal', ylabel='AQI'>



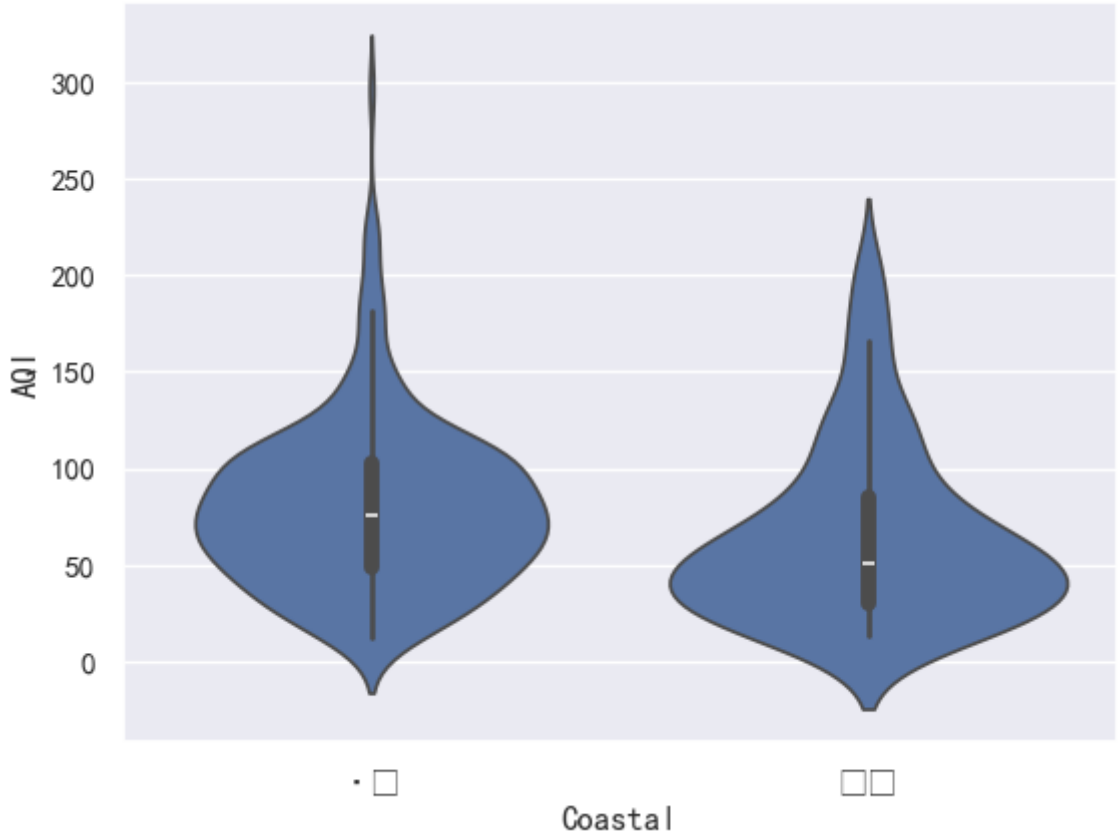
In [121…
```
sns.boxplot(x = "Coastal" , y = "AQI" , data = data)
```

Out[121… <Axes: xlabel='Coastal', ylabel='AQI'>



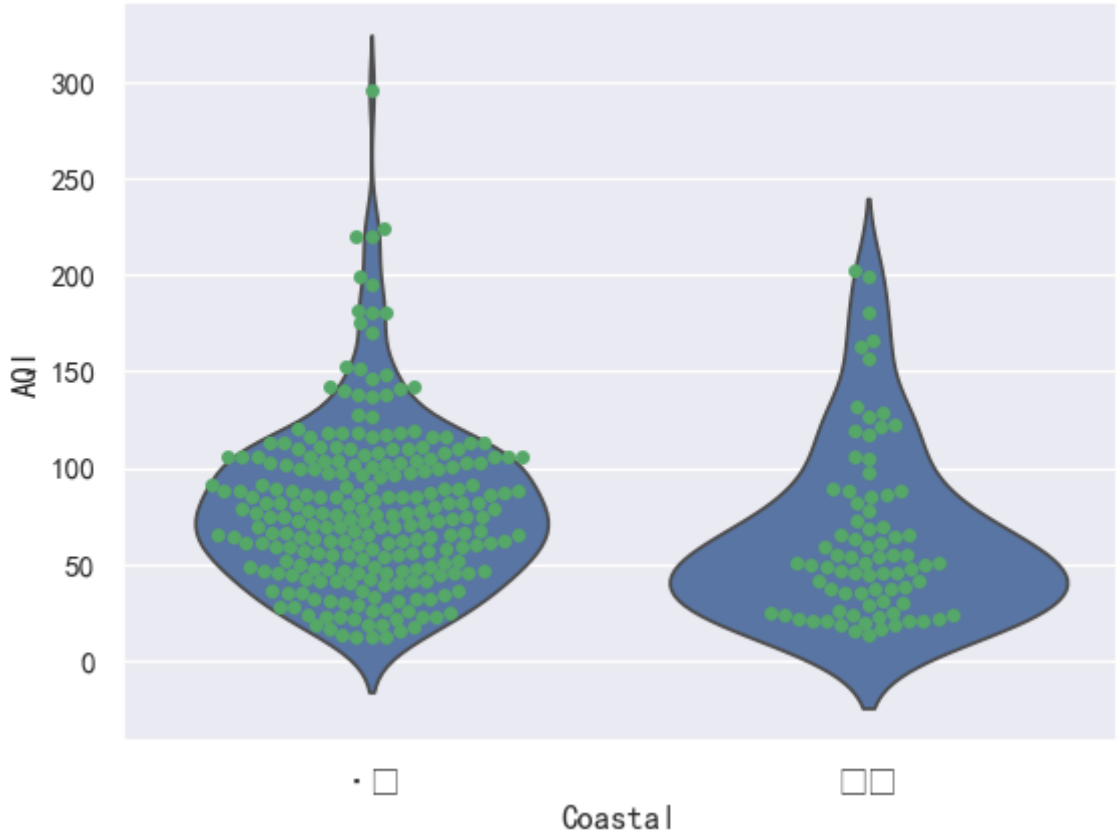
In [123…
```
sns.violinplot(x = "Coastal" , y = "AQI" , data = data)
```

Out[123…
```
<Axes: xlabel='Coastal', ylabel='AQI'>
```



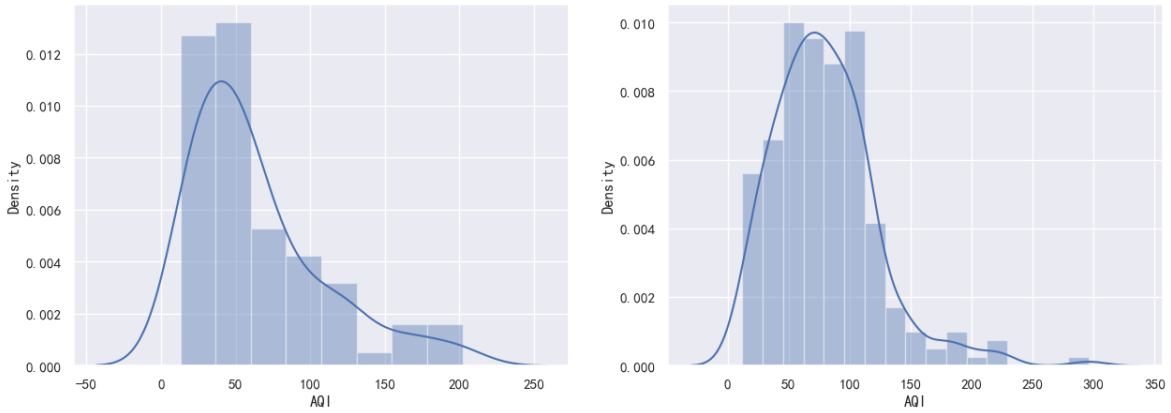
In [125…
```
sns.violinplot(x = "Coastal" , y = "AQI" , data = data ,inner = None)
sns.swarmplot(x = "Coastal", y = "AQI", color = "g" , data = data)
```

Out[125… <Axes: xlabel='Coastal', ylabel='AQI'>



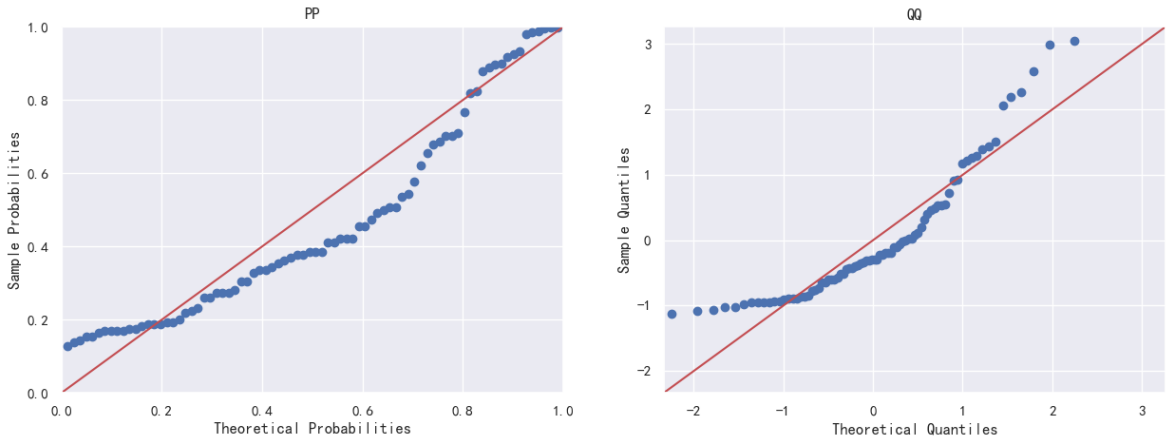
In [159…
```
from scipy import stats
coastal = data[data["Coastal"] == "是"]["AQI"]
inland  = data[data["Coastal"] == "否"]["AQI"]

fig , ax = plt.subplots(1,2)
fig.set_size_inches(15,5)

sns.distplot(coastal , ax =ax[0])
sns.distplot(inland  , ax =ax[1])
```

Out[159… <Axes: xlabel='AQI', ylabel='Density'>

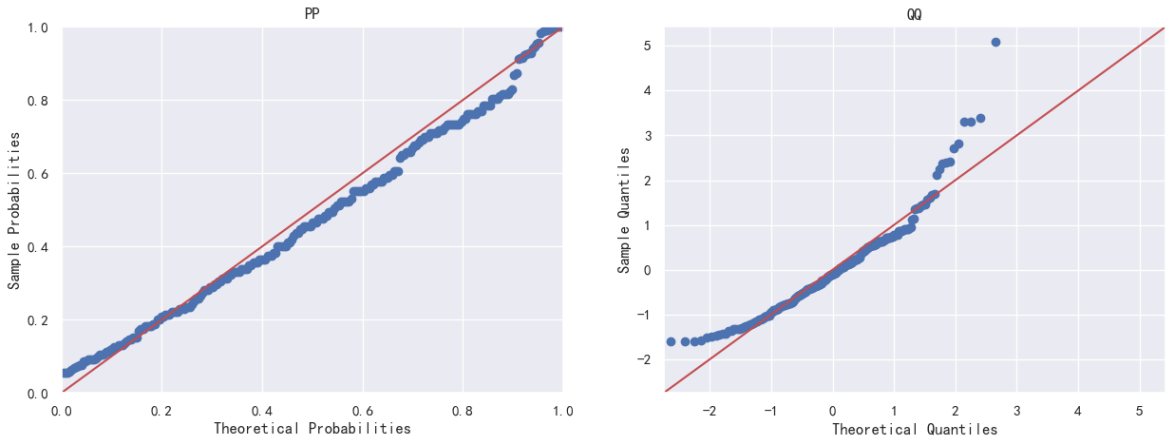


In [161…
```
import statsmodels.api as sm

def plot_pp_qq(d):
#     """绘制PP图与QQ图的函数
#     Parameters
#     ----------
#     d:array-like
```

```
#         要绘制的数值
#     """
    fig , ax = plt.subplots(1,2)
    fig.set_size_inches(15,5)
    scale_data = (d - d.mean()) / d.std()
    #创建ProbPlot对象，用于绘制PP图与QQ图
    #data：样本数据。
    #dist：分布，默认为正态分布。数据data会与该分布进行对比。

    p = sm.ProbPlot(data = scale_data , dist = stats.norm)
    p.ppplot(line = "45" , ax = ax[0])
    ax[0].set_title("PP")
    p.qqplot(line = "45" , ax = ax[1])
    ax[1].set_title("QQ")
    plt.show()

plot_pp_qq(coastal)
```



In [157…
```
plot_pp_qq(inland)
```



In [163…
```
print(stats.normaltest(coastal))
print(stats.normaltest(inland))
```

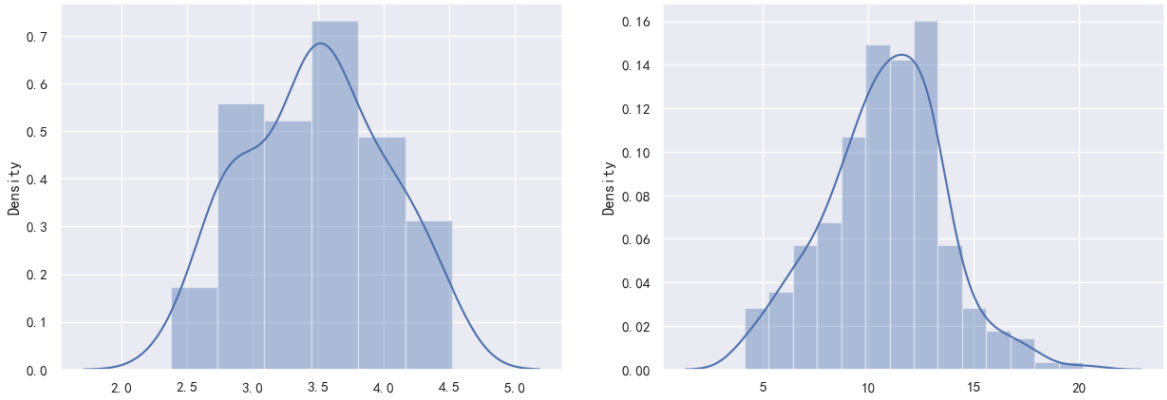
```
NormaltestResult(statistic=21.143187601170315, pvalue=2.5633927252643683e-05)
NormaltestResult(statistic=67.05204155603354, pvalue=2.7531772733964943e-15)
```

In [165…
```
# Box-Cox转换
bc_coastal , _ = stats.boxcox(coastal)
bc_inland , _ = stats.boxcox(inland)
```

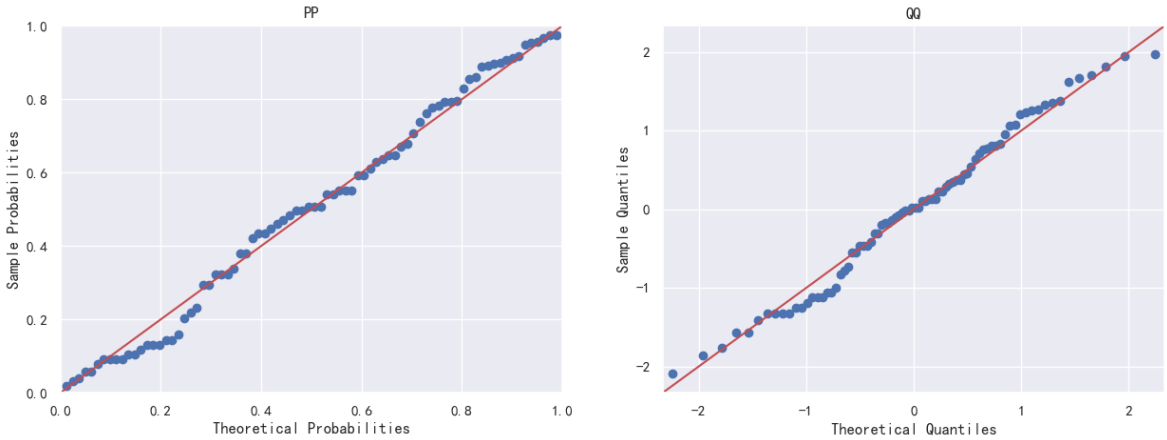
In [167…
```
# 正态分布检验（二次）
fig , ax = plt.subplots(1,2)
```

```
fig.set_size_inches(15,5)

# 两个样本的分布
sns.distplot(bc_coastal , ax = ax[0])
sns.distplot(bc_inland  , ax = ax[1])
```

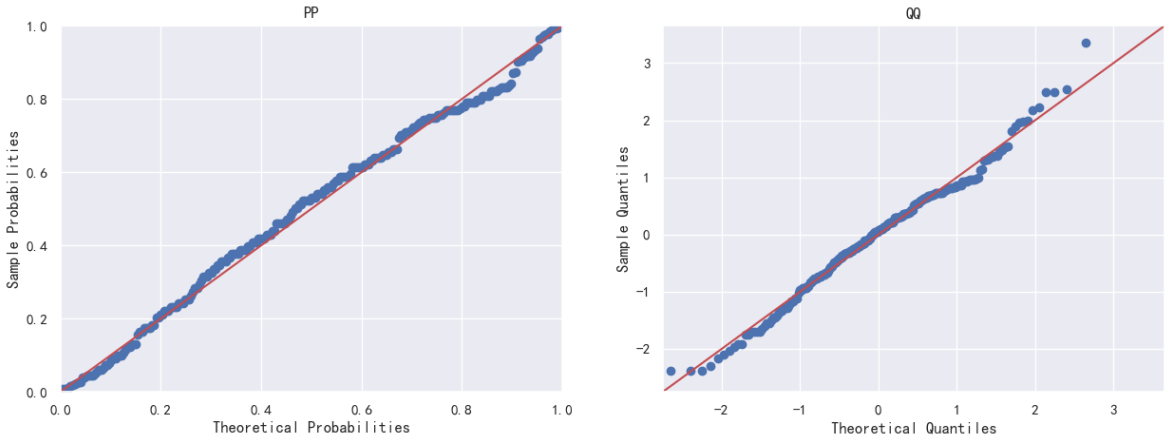
Out[167… <Axes: ylabel='Density'>



In [169…
```
plot_pp_qq(bc_coastal)
```



In [171…
```
plot_pp_qq(bc_inland)
```



In [173…
```
print(stats.normaltest(bc_coastal))
print(stats.normaltest(bc_inland))
```

```
NormaltestResult(statistic=4.306699215273687, pvalue=0.11609463428866755)
NormaltestResult(statistic=0.9419430390414003, pvalue=0.624395361237339)
```

In [175…
```
# 两独立样本t检验
```

In [177…
```
stats.levene(bc_coastal , bc_inland)
```

Out[177…
```
LeveneResult(statistic=79.42938159164872, pvalue=3.7402960221188775e-17)
```

In [179…
```
r = stats.ttest_ind(bc_coastal , bc_inland , equal_var = False)
print(r)
```

```
TtestResult(statistic=-38.63740276503151, pvalue=2.1016305682842884e-116, df=290.
104035826179)
```

In [181…
```
p = stats.t.sf(r.statistic ,df = len(coastal) + len(inland) - 2)
print(p)
```

```
1.0
```

In [183…
```
# 非参数检验
print(stats.mannwhitneyu(coastal , inland))

print(stats.ranksums(coastal , inland))
```

```
MannwhitneyuResult(statistic=7128.0, pvalue=0.00025091777560515467)
RanksumsResult(statistic=-3.661763076284923, pvalue=0.00025048546022636835)
```

In [185…
```
# 近似使用Z检验
stats.levene(coastal , inland)
```

Out[185…
```
LeveneResult(statistic=0.03818483157315866, pvalue=0.8451953286335752)
```

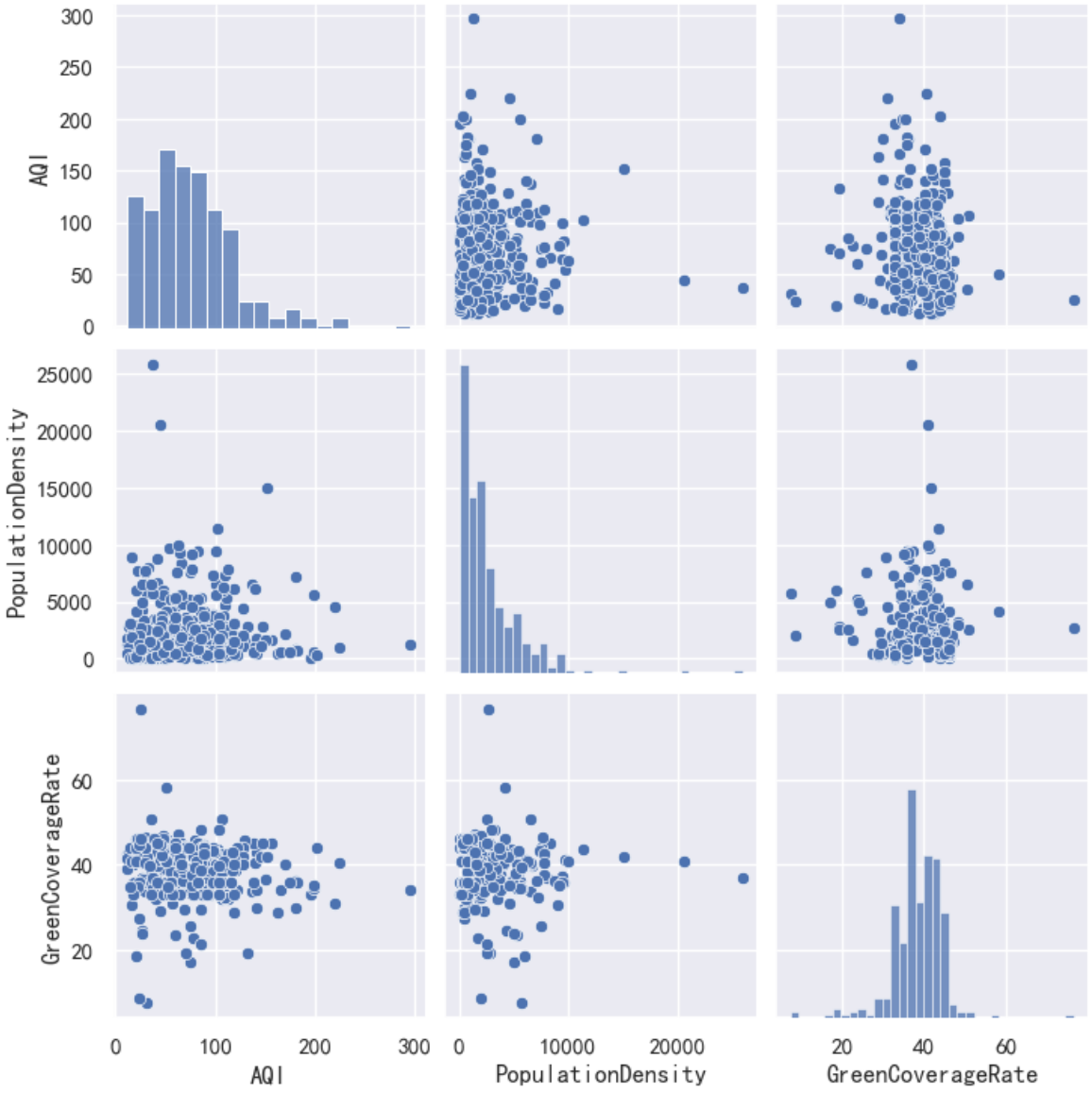
In [187…
```
r = stats.ttest_ind(coastal , inland ,equal_var = True)
print(r)

p = stats.t.sf(r.statistic ,df = len(coastal) + len(inland) - 2)
print(p)
```

```
TtestResult(statistic=-2.804017387113549, pvalue=0.005352397281668703, df=323.0)
0.9973238013591657
```

空气质量主要受哪些因素影响

In [189…
```
# 变量较多，图像就会变的比较小，索引仅选择部分进行绘制散点图矩阵
sns.pairplot(data , vars = ["AQI" , "PopulationDensity" ,
                            "GreenCoverageRate"])
```

Out[189…
```
<seaborn.axisgrid.PairGrid at 0x1bbeeeaf560>
```



In [191…
```
x = data["AQI"]
y = data["Precipitation"]
print(x - x.mean())
type(y)
# print(y - y.mean())
```

```
0     -52.809231
1      61.190769
2       9.190769
3     -47.809231
4       3.190769
          ...
320     3.190769
321    10.190769
322    40.190769
323    42.190769
324   -15.809231
Name: AQI, Length: 325, dtype: float64
```

Out[191… pandas.core.series.Series

In [213…
```
# x = data["AQI"]
# y = data["Precipitation"]

# a = (x - x.mean())*(y - y.mean())
```

```
# cov = np.sum(a) / (len(a) - 1)
# print("协方差:" , x.cov(y))

# corr = cov / np.sqrt(x.var()*y.var())
# print("相关系数: " , corr)

# # print("协方差:" , x.cov(y))
# # print("相关系数: ", corr(y))
```

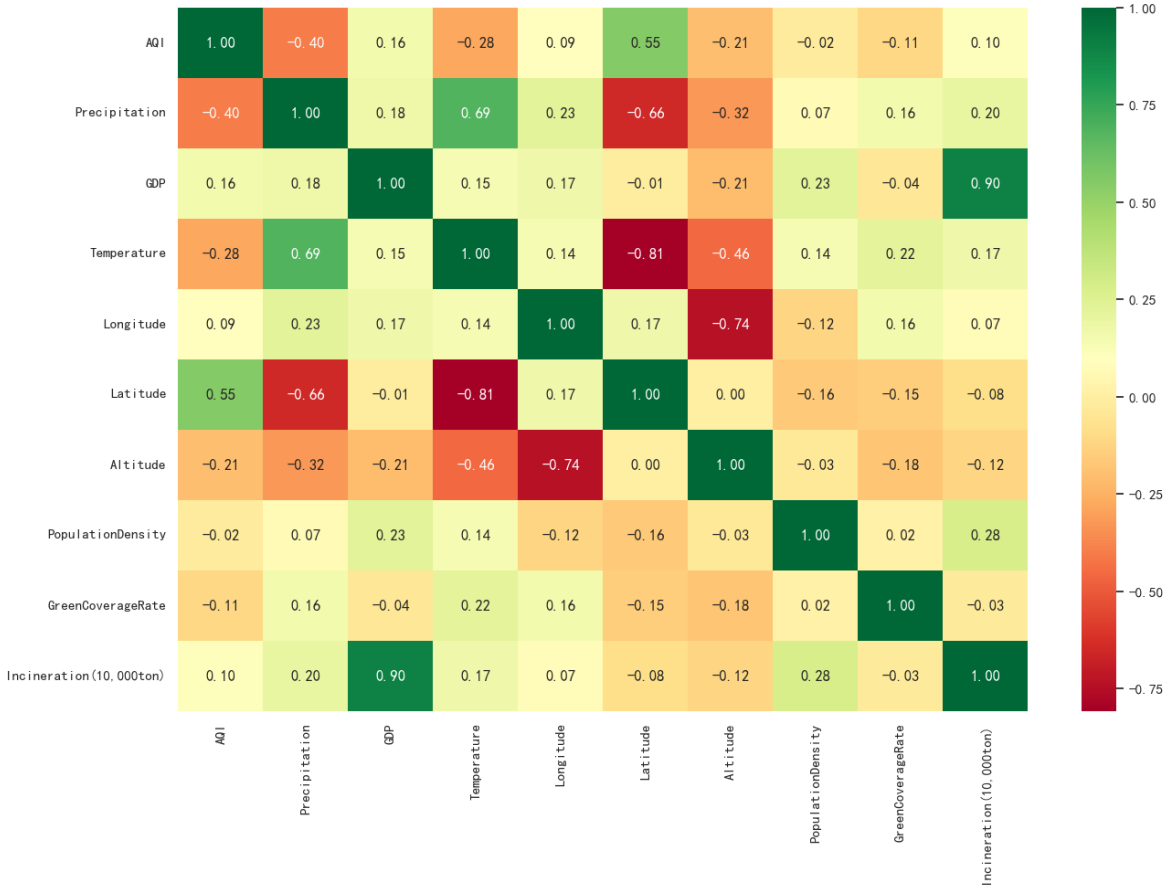
In [201…

```
# 只选择数值型列
numeric_data = data.select_dtypes(include=['number'])
correlation_matrix = numeric_data.corr()
print(correlation_matrix)
```

```
                              AQI  Precipitation       GDP  Temperature  \
AQI                      1.000000      -0.404378  0.158404    -0.284551
Precipitation           -0.404378       1.000000  0.175849     0.686503
GDP                      0.158404       0.175849  1.000000     0.145554
Temperature             -0.284551       0.686503  0.145554     1.000000
Longitude                0.094299       0.225373  0.173037     0.140873
Latitude                 0.553932      -0.658193 -0.010074    -0.806979
Altitude                -0.206416      -0.323631 -0.208350    -0.458015
PopulationDensity       -0.018488       0.065860  0.227883     0.143276
GreenCoverageRate       -0.108193       0.158939 -0.038672     0.217958
Incineration(10,000ton)  0.104481       0.200696  0.899550     0.173594

                          Longitude  Latitude  Altitude  PopulationDensity  \
AQI                        0.094299  0.553932 -0.206416          -0.018488
Precipitation              0.225373 -0.658193 -0.323631           0.065860
GDP                        0.173037 -0.010074 -0.208350           0.227883
Temperature                0.140873 -0.806979 -0.458015           0.143276
Longitude                  1.000000  0.173903 -0.737326          -0.121305
Latitude                   0.173903  1.000000  0.000449          -0.163498
Altitude                  -0.737326  0.000449  1.000000          -0.033568
PopulationDensity         -0.121305 -0.163498 -0.033568           1.000000
GreenCoverageRate          0.155278 -0.146780 -0.179219           0.016873
Incineration(10,000ton)    0.072024 -0.081629 -0.121599           0.281934

                          GreenCoverageRate  Incineration(10,000ton)
AQI                               -0.108193                 0.104481
Precipitation                      0.158939                 0.200696
GDP                               -0.038672                 0.899550
Temperature                        0.217958                 0.173594
Longitude                          0.155278                 0.072024
Latitude                          -0.146780                -0.081629
Altitude                          -0.179219                -0.121599
PopulationDensity                  0.016873                 0.281934
GreenCoverageRate                  1.000000                -0.028239
Incineration(10,000ton)           -0.028239                 1.000000
```

In [205…

```
# 画热图
plt.figure(figsize=(15, 10))
ax = sns.heatmap(correlation_matrix, cmap=plt.cm.RdYlGn, annot=True, fmt=".2f")
plt.show()
```



空气质量验证

In [207…
```
data["AQI"].mean( )
```

Out[207…
```
75.80923076923077
```

In [209…
```
r = stats.ttest_1samp(data["AQI"] , 75)
print(r)
# print("t值: " , r.statistic)
# print("p值: " , r.pvalue)
```

```
TtestResult(statistic=0.33452058395061307, pvalue=0.7382032031274794, df=324)
```

In [211…
```
mean = data["AQI"].mean()
std = data["AQI"].std()
stats.t.interval(0.95 , df = len(data) - 1 , loc = mean , scale = std / np.sqrt(
```

Out[211…
```
(71.05015134649922, 80.56831019196231)
```

In [ ]: