# AQI分析与预测

## 背景信息

AQI(Air Quality Index)，指空气质量指数，用来衡量空气清洁或污染的程度。值越小，表示空气质量越好。近年来，因为环境问题，空气质量也越来越受到人们的重视。

## 任务说明与知识要点

我们期望能够运用数据分析的相关技术，对全国城市空气质量进行研究与分析，希望能够解决如下疑问：

- 哪些城市的空气质量较好 / 较差？【描述性统计分析】
- 空气质量在地理位置分布上，是否具有一定的规律性？【描述性统计分析】
- 临海城市的空气质量是否有别于内陆城市？【推断统计分析】
- 空气质量主要受哪些因素影响？【相关系数分析】
- 全国城市空气质量普遍处于何种水平？【区间估计】
- 怎样预测一个城市的空气质量？【统计建模】

## 数据集描述

我们现在获取了2015年空气质量指数集。该数据集包含全国主要城市的相关数据以及空气质量指数。

| 列名 | 含义 |
|---|---|
| City | 城市名 |
| AQI | 空气质量指数 |
| Precipitation | 降雨量 |
| GDP | 城市生产总值 |
| Temperature | 温度 |
| Longitude | 经度 |
| Latitude | 纬度 |
| Altitude | 海拔高度 |
| PopulationDensity | 人口密度 |
| Coastal | 是否沿海 |
| GreenCoverageRate | 绿化覆盖率 |
| Incineration(10,000ton) | 焚烧量（10000吨） |

# 程序实现

## 导入相关的库

导入需要的库，同时，进行一些初始化的设置。

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import warnings
import seaborn as sns

sns.set(style="darkgrid", font="SimHei", rc={"axes.unicode_minus": False})
# mpl.rcParams["font.family"] = "SimHei"
# mpl.rcParams["axes.unicode_minus"] = False

warnings.filterwarnings("ignore")
```

## 加载相关的数据集

- 加载相关的数据集。
- 可以使用head / tail / sample查看数据的大致情况。

In [2]:
```
data = pd.read_csv("data.csv")
print(data.shape)
data.head()
```

```
(325, 12)
```

Out[2]:

| | City | AQI | Precipitation | GDP | Temperature | Longitude | Latitude | Altitude | PopulationDensity | Coastal | GreenCoverageRate | Incineration(10,000ton) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Ngawa Prefecture | 23 | 665.1 | 271.13 | 8.200000 | 102.224650 | 31.899410 | 2617.0 | 11 | 0 | 36.00 | 23.00 |
| 1 | Aksu City | 137 | 80.4 | 610.00 | 12.276712 | 80.263380 | 41.167540 | 1108.0 | 6547 | 0 | 33.94 | 23.00 |
| 2 | Alxa League | 85 | 150.0 | 322.58 | 24.200000 | 105.728950 | 38.851920 | 1673.0 | 1 | 0 | 36.00 | 23.00 |
| 3 | Ngari | 28 | 74.2 | 37.40 | 1.000000 | 80.105800 | 32.501110 | 4280.0 | 1 | 0 | 36.00 | 23.00 |
| 4 | Anqin City | 79 | 2127.8 | 1613.20 | 17.291781 | 117.034431 | 30.512646 | 13.0 | 2271 | 0 | 45.80 | 27.48 |

## 数据清洗

### 缺失值处理

我们可以使用如下方法查看缺失值：

- info
- isnull

In [3]:
```
data.info()
# data.isnull().sum(axis=0)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 325 entries, 0 to 324
Data columns (total 12 columns):
City                        325 non-null object
AQI                         325 non-null int64
Precipitation               321 non-null float64
GDP                         325 non-null float64
Temperature                 325 non-null float64
Longitude                   325 non-null float64
Latitude                    325 non-null float64
```

```
Altitude                  325 non-null float64
PopulationDensity         325 non-null int64
Coastal                   325 non-null int64
GreenCoverageRate         325 non-null float64
Incineration(10,000ton)   325 non-null float64
dtypes: float64(8), int64(3), object(1)
memory usage: 30.5+ KB
```

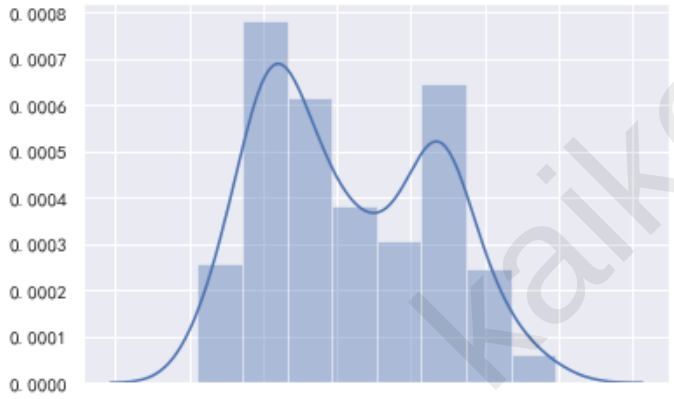
通过观察得知，降雨量（Precipitation）列存在少许缺失值，需要我们进行处理。


如果降雨量这一列中，有100条记录存在缺失值，我们如何处理会更好些？
A 删除缺失值所在的行（记录）。
B 使用均值进行填充。
C 使用中值进行填充。
D B或C。
E 暂时还无法判断。

In [4]:
```
print(data["Precipitation"].skew())
sns.distplot(data["Precipitation"].dropna())
```

```
0.27360760671177387
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x58d8b70>



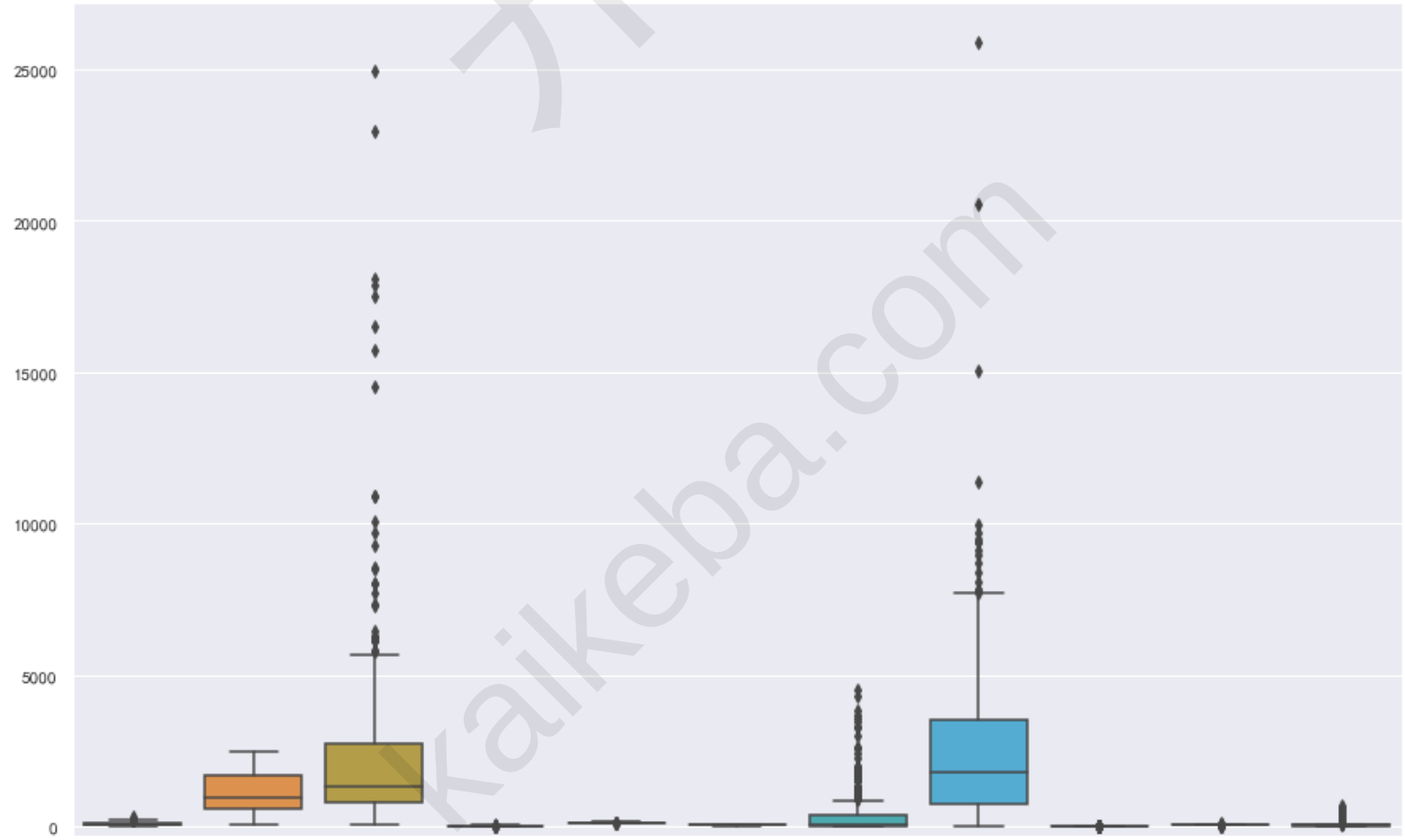

In [5]:
```
data.fillna({"Precipitation": data["Precipitation"].median()}, inplace=True)
```

## 异常值处理

- 通过describe查看数值信息。
- 可配合箱线图辅助。

In [6]:
```
# data.describe()
plt.figure(figsize=(15, 10))
plt.xticks(rotation=45, fontsize=15)
sns.boxplot(data=data)
```

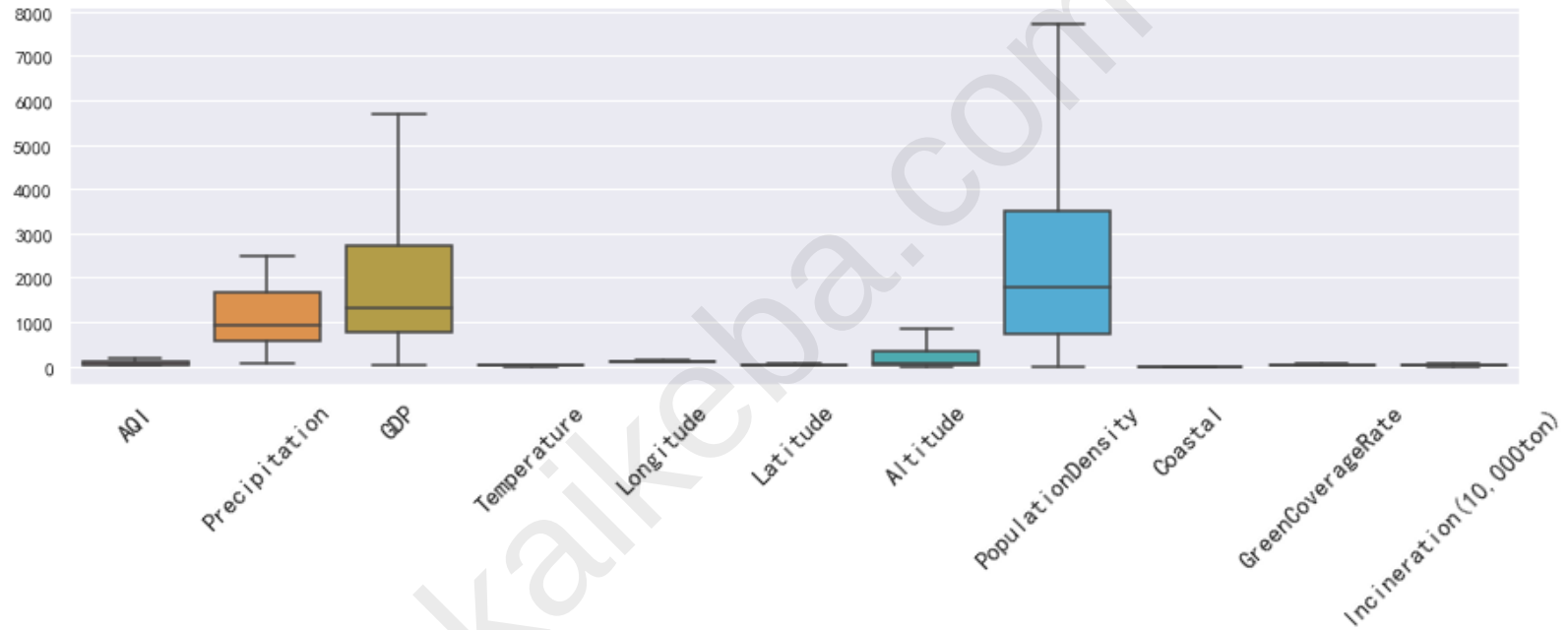
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0xb800ef0>




In [7]:
```
t = data.copy()
for k in t:
    if pd.api.types.is_numeric_dtype(t[k]):
        o = t[k].describe()
        IQR = o["75%"] - o["25%"]
        lower = o["25%"] - 1.5 * IQR
        upper = o["75%"] + 1.5 * IQR
        t[k][t[k] < lower] = lower
        t[k][t[k] > upper] = upper
```

In [8]:
```
plt.figure(figsize=(15, 4))
plt.xticks(rotation=45, fontsize=15)
sns.boxplot(data=t)
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0xbc5de48>



### 重复值处理

- 使用duplicate检查重复值。可配合keep参数进行调整。
- 使用drop_duplicate删除重复值。

In [9]:
```
# 发现重复值。
data.duplicated().sum()
# 查看哪些记录出现了重复值。
# data[data.duplicated()]
# 删除重复值。
data.drop_duplicates(inplace=True)
```

# 数据分析

## 空气质量最好 / 最差的5个城市。

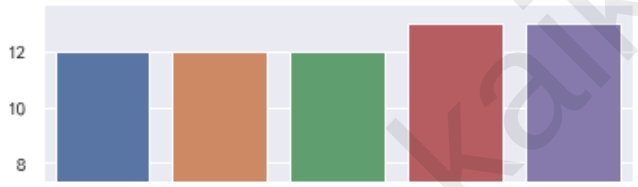
空气质量的好坏可以为我们以后选择工作，旅游等地提供参考。

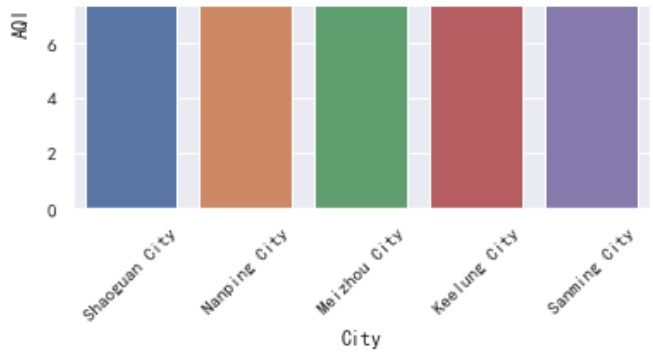
#### 最好的5个城市

In [10]:
```
t = data[["City", "AQI"]].sort_values("AQI")
display(t.iloc[:5])
plt.xticks(rotation=45)
sns.barplot(x="City", y="AQI", data=t.iloc[:5])
```

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

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xb924ef0>
```






我们发现，空气质量最好的5个城市为：

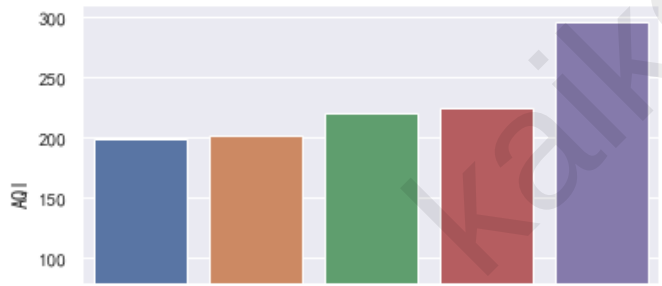
1. 韶关市
2. 南平市
3. 梅州市
4. 基隆市
5. 三明市

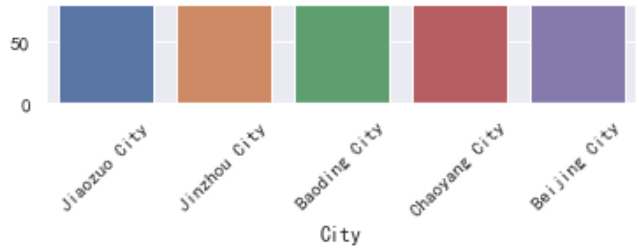
**最差的5个城市**

In [11]:
```
display(t.iloc[-5:])
plt.xticks(rotation=45)
sns.barplot(x="City", y="AQI", data=t.iloc[-5:])
```

| | City | AQI |
|---|---|---|
| **105** | Jiaozuo City | 199 |
| **112** | Jinzhou City | 202 |
| **13** | Baoding City | 220 |
| **26** | Chaoyang City | 224 |
| **16** | Beijing City | 296 |

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0xbaa3198>






我们得出空气质量最差的5个城市为：

1. 北京市
2. 朝阳市
3. 保定市
4. 锦州市
5. 焦作市

## 全国城市的空气质量

### 城市空气质量等级统计

国家对空气质量进行等级划分，划分标准如下表所示：

| AQI指数 | 等级 | 描述 |
|---|---|---|
| 0-50 | 一级 | 优 |
| 51-100 | 二级 | 良 |
| 101-150 | 三级 | 轻度污染 |
| 151-200 | 四级 | 中度污染 |
| 201-300 | 五级 | 重度污染 |
| >300 | 六级 | 严重污染 |

根据该标准，我们来统计下，全国空气质量每个等级的数量。

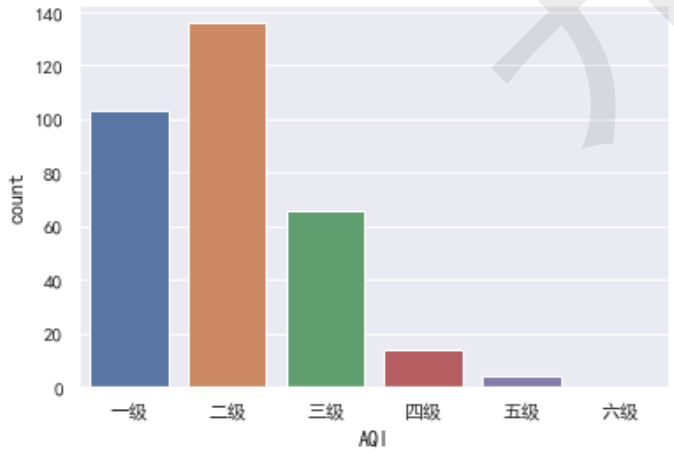
In [12]:
```
# 编写函数，将AQI转换为对应的等级。
def value_to_level(AQI):
    if AQI >= 0 and AQI <= 50:
        return "一级"
    elif AQI >= 51 and AQI <= 100:
        return "二级"
    elif AQI >= 101 and AQI <= 150:
        return "三级"
```

```
    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)
display(level.value_counts())
sns.countplot(x=level, order=["一级", "二级", "三级", "四级", "五级", "六级"])
```

```
二级    136
一级    103
三级     66
四级     14
五级      4
Name: AQI, dtype: int64
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0xbb1a7b8>



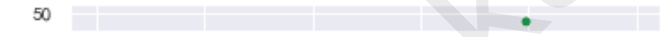

可见，我们城市的空气质量主要以一级（优）与二级（良）为主，三级（轻度污染）占一部分，更高污染的城市占少数。

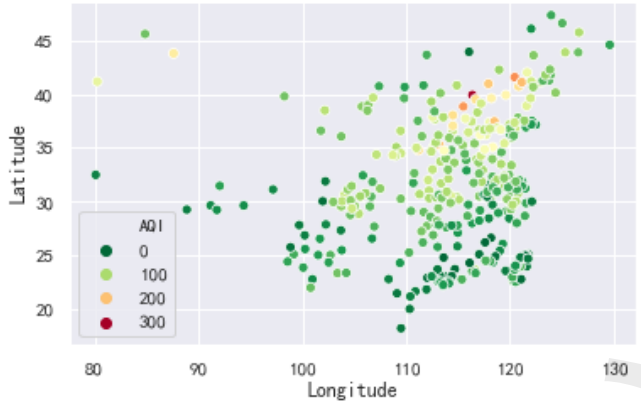
**空气质量指数分布**

我们来绘制一下全国各城市的空气质量指数分布图。

In [13]:
```
sns.scatterplot(x="Longitude", y="Latitude", hue="AQI", palette=plt.cm.RdYlGn_r, data=data)
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0xbbd8e10>






从结果我们可以发现，从大致的地理位置上看，西部城市好于东部城市，南部城市好于北部城市。

## 关于空气质量的验证

江湖传闻，全国所有城市的空气质量指数均值在72左右，请问，这个消息可靠吗？

城市平均空气质量指数，我们可以很容易的进行计算。

In [14]:
```
data["AQI"].mean()
```

Out[14]: 75.3343653250774


我们计算的值大于传闻值72，因此，我们认为，江湖传闻实属一派胡言，不可尽信。请问这样认为正确吗？

A 正确

B 不正确

首先，我们要清楚，江湖传闻的，是全国所有城市的平均空气质量指数，而我们统计的，只是所有城市中的一部分抽样而已。因此，我们一次抽样统计的均值，并不能代表总体（所有城市）的均值。

要弄清江湖传闻是否可靠，最直接有效的方式，就是将全国所有的城市的空气质量指数都测量一下，然后进行求均值。然而，这是非常繁重且不现实的任务。因此，可行的方案是，我们从全国所有城市中进行抽样，使用抽样的均值来估计总体的均值。

**中心极限定理**

如果总体（分布不重要）均值为$\mu$，方差为$\sigma^2$，我们进行随机抽样，样本容量为$n$，当$n$增大时，则样本均值逐渐趋近服从正态分布：$\bar{X} \sim N(\mu, \sigma^2/n)$。

我们可以得到如下结论：

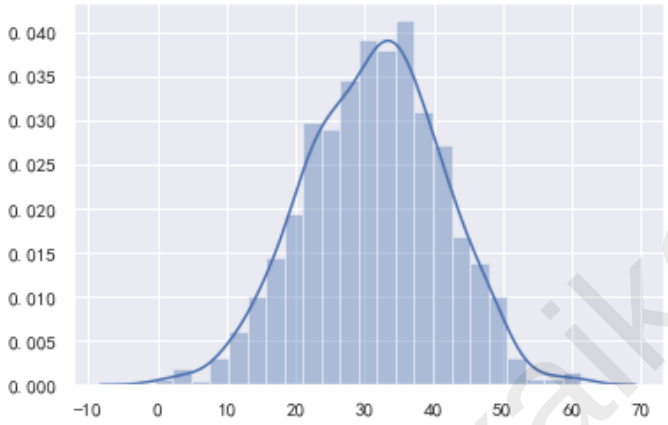
1. 进行多次抽样，则每次抽样会得到一个均值，这些均值会围绕在总体均值左右，呈正态分布。
2. 当样本容量$n$足够大时，样本均值服从正态分布。
    - 样本均值构成的正态分布，其均值等于总体均值$\mu$。
    - 样本均值构成的正态分布，其标准差等于总体标准差$\sigma$除以$\sqrt{n}$。

说明：样本均值分布的标准差，我们称为标准误差，简称标准误。

In [15]:
```
# 定义总体数据。
all_ = np.random.normal(loc=30, scale=80, size=10000)
# 创建均值数组。
mean_arr = np.zeros(1000)
for i in range(len(mean_arr)):
    mean_arr[i] = np.random.choice(all_, size=64, replace=False).mean()
print("样本均值：", mean_arr.mean())
print("样本标准差：", mean_arr.std())
print("偏度：", pd.Series(mean_arr).skew())
sns.distplot(mean_arr)
```

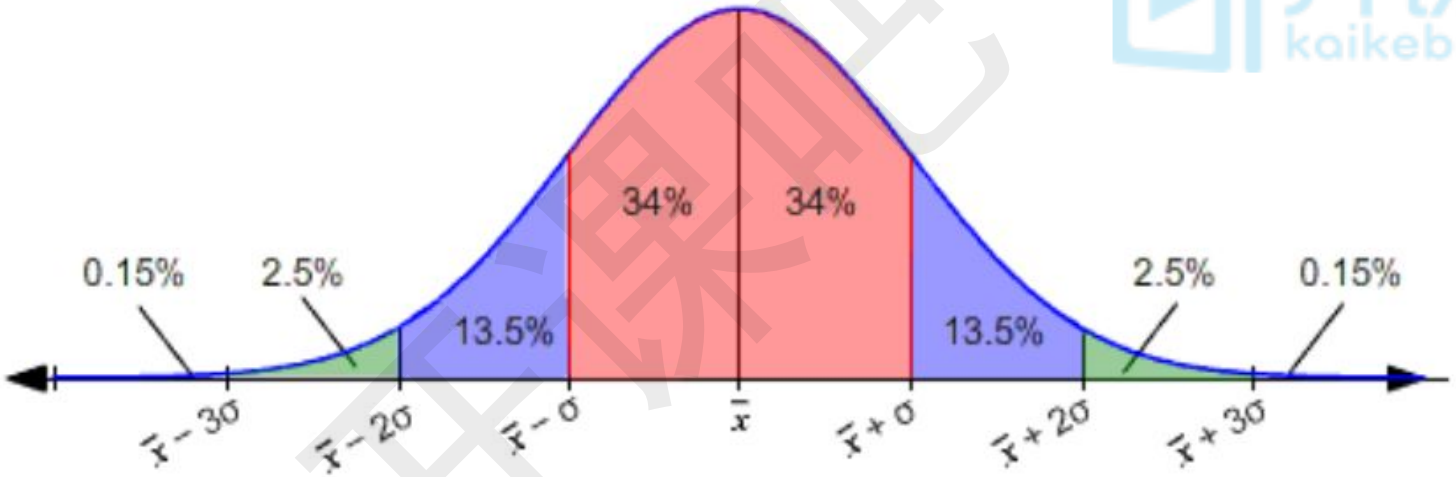
```
样本均值： 31.351438228008604
样本标准差： 9.87330575446937
偏度： -0.08290430384690746
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xbad1cf8>
```




**置信区间**

接下来，我们现在根据正态分布的特性，进行概率上的统计，如下图：




在正态分布中，数据的分布比例如下：

- 以均值为中心，在一倍标准差内$(\bar{x}-\sigma, \bar{x}+\sigma)$，包含约68%的样本数据。
- 以均值为中心，在二倍标准差内$(\bar{x}-2\sigma, \bar{x}+2\sigma)$，包含约95%的样本数据。
- 以均值为中心，在三倍标准差内$(\bar{x}-3\sigma, \bar{x}+3\sigma)$，包含约99.7%的样本数据。

In [16]:
```
# 定义标准差
scale = 50
# 定义数据。
x = np.random.normal(0, scale, size=100000)
# 定义标准差的倍数，倍数从1到3。
for times in range(1, 4):
    y = x[(x >= -times * scale) & (x <= times * scale)]
    print(f"{times}倍标准差：")
    print(f"{len(y) * 100 / len(x)}%")
```

```
1倍标准差：
68.051%
2倍标准差：
95.468%
3倍标准差：
99.76%
```

根据中心极限定理，如果多次抽样，则样本均值构成的正态分布。如果我们对总体进行一次抽样，则本次抽样个体的均值有95%的概率会在二倍标准差内，仅有5%的概率会在二倍标准差外。根据小概率事件（很小的概率在一次抽样中基本不会发生），如果抽样的个体均值落在二倍标准差之外，我们就可以认为，本次抽

样来自的总体，该总体的均值并非是我们所期望的均值。

通常，我们以二倍标准差作为判定依据，则以均值为中心，正负二倍标准差构成的区间，就是置信区间。而二倍标准差包含了95%的数据，因此，此时的置信度为95%。换言之，我们有信心认为，总体的均值有95%的概率会在置信区间之内。

**假设检验——t检验**

假设检验，其目的是通过收集到的数据，来验证某个假设是否成立。在假设检验中，我们会建立两个完全对立的假设，分别为原假设（零假设）$H_0$与备则假设（对立假设）$H_1$。然后根据样本信息进行分析判断，得出$P$值（概率值）。

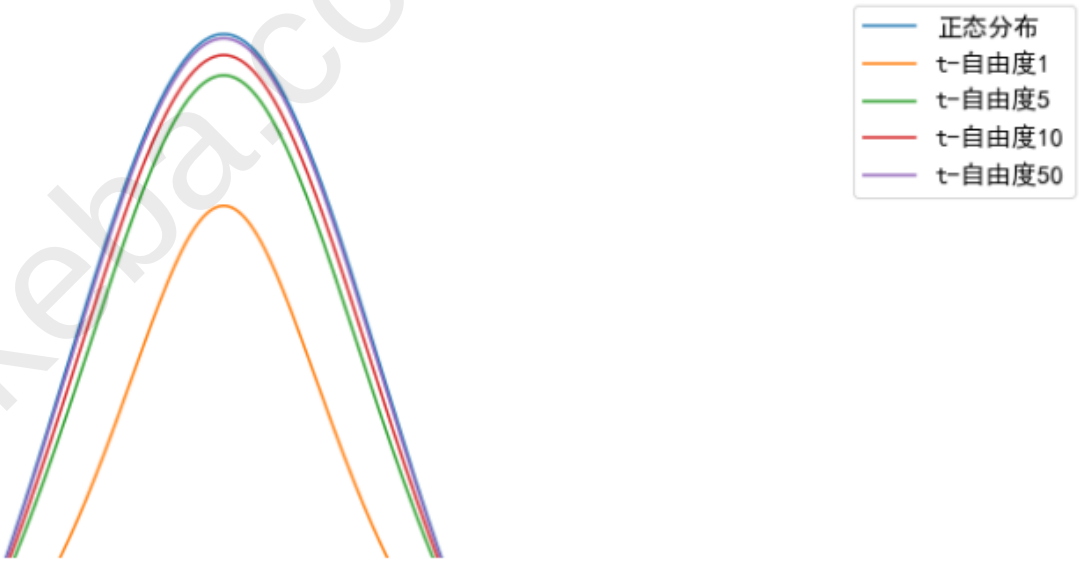
假设检验基于小概率反证法，即我们认为小概率事件在一次试验中是不会发生的。如果小概率事件发生，则我们就拒绝原假设，而接受备择假设。否则，我们就没有充分的理由推翻原假设，此时，我们选择去接受原假设。

t检验，就是假设检验的一种，可以用来检验一次抽样中样本均值与总体均值的比较（二者差异是否显著）。其计算方式如下：

$$t = \frac{\bar{x}-\mu_0}{S_{\bar{x}}} = \frac{\bar{x}-\mu_0}{S/\sqrt{n}}$$

- $\bar{x}$为一次抽样中，所有个体的均值。
- $\mu_0$为待检验的均值。
- $S_{\bar{x}}$为样本均值的标准差（标准误差）。
- $S$为一次抽样中，个体的标准差。
- $n$为样本容量。

$t$统计量服从$t$分布，当自由度（样本容量 - 1）逐渐增大时，$t$分布近似于正态分布。



In [17]:
```
from scipy import stats

r = stats.ttest_1samp(data["AQI"], 72)
print("t值: ", r.statistic)
print("p值: ", r.pvalue)
```

```
t值:  1.393763441074581
p值:  0.16435019471704654
```

我们可以看到，$P$值大于0.05，故在显著度水平为0.05检验下，我们无法拒绝原假设，因此接受原假设。
同样，我们现在可以来计算下，全国所有城市平均空气质量指数的置信区间。

In [18]:
```
n = len(data)
df = n - 1
left = stats.t.ppf(0.025, df=df)
right = stats.t.ppf(0.975, df=df)
print(left, right)
mean = data["AQI"].mean()
std = data["AQI"].std()
mean + left * (std / np.sqrt(n)), mean + right * (std / np.sqrt(n))
```

```
-1.9673585853224684 1.967358585322468
```

Out[18]:
```
(70.6277615675309, 80.0409690826239)
```

由此，我们就计算出全国所有城市平均空气质量指数所在的置信区间，大致在70.63 ~ 80.04之间，置信度为95%。

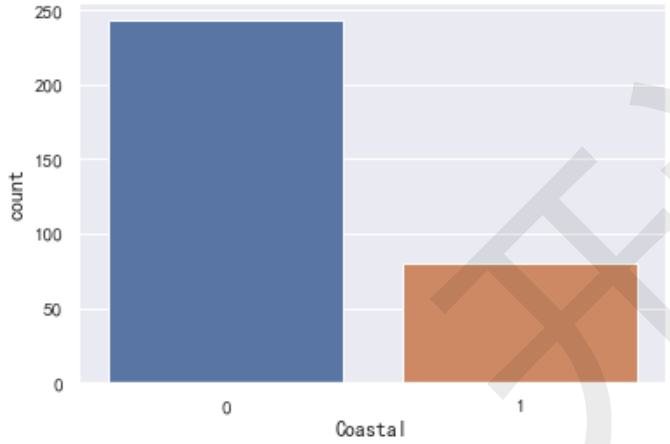
## 临海城市是否空气质量优于内陆城市？

我们首先来统计下临海城市与内陆城市的数量。

In [19]:
```
display(data["Coastal"].value_counts())
sns.countplot(x="Coastal", data=data)
```

```
0    243
1     80
Name: Coastal, dtype: int64
```

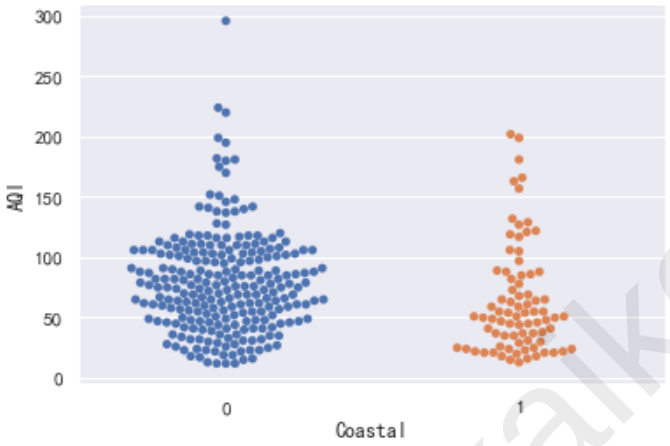
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0xc15e048>




然后，我们来观察一下临海城市与内陆城市的散点分布。

In [20]:
```
sns.swarmplot(x="Coastal", y="AQI", data=data)
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0xc1a7a58>

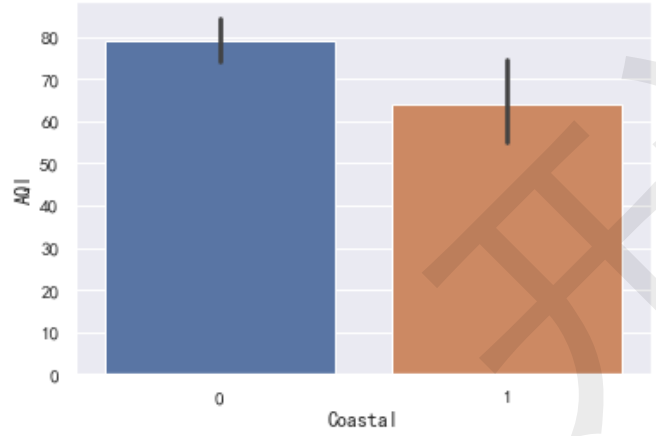



然后，我们再来分组计算空气质量的均值。

In [21]:
```
display(data.groupby("Coastal")["AQI"].mean())
sns.barplot(x="Coastal", y="AQI", data=data)
```

```
Coastal
0    79.045267
1    64.062500
Name: AQI, dtype: float64
```

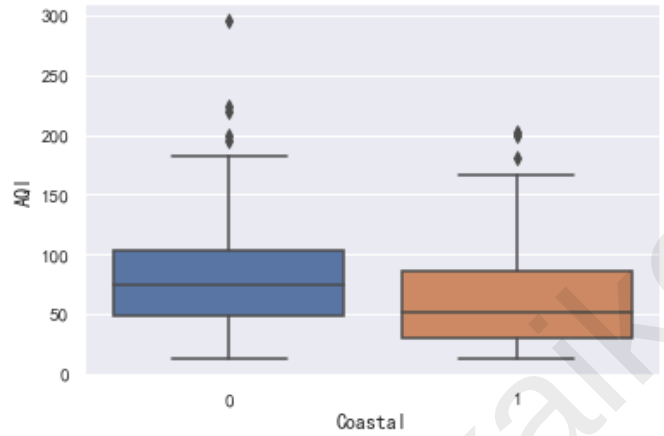
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0xc201cf8>




在柱形图中，仅显示了内陆城市与临海城市空气质量指数（AQI）的均值对比，我们可以使用箱线图来显示更多的信息。

In [22]:
```
sns.boxplot(x="Coastal", y="AQI", data=data)
```

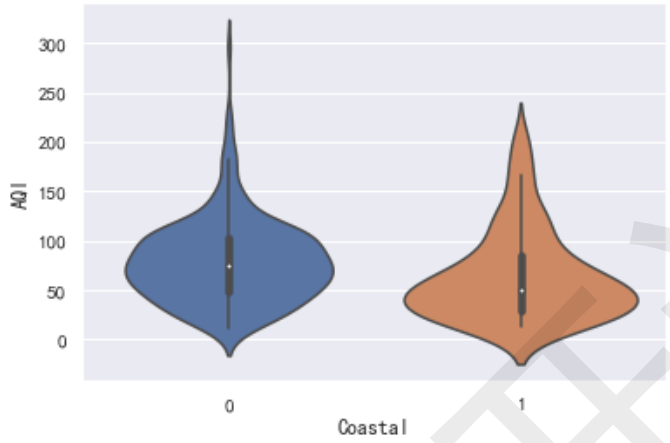
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0xc255860>




我们也可以绘制小提琴图，除了能够展示箱线图的信息外，还能呈现出分布的密度。

In [23]:
```
sns.violinplot(x="Coastal", y="AQI", data=data)
```

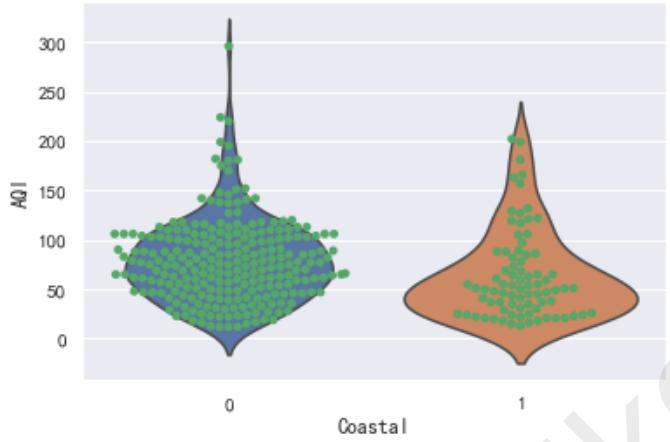
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0xc2cee80>




我们可以将散点与箱线图或小提琴图结合在一起进行绘制，下面以小提琴图为例。

In [24]:
```
sns.violinplot(x="Coastal", y="AQI", data=data, inner=None)
sns.swarmplot(x="Coastal", y="AQI", color="g", data=data)
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0xc31aa58>




## 课堂练习

至此，我们可以得出什么结论？
A 沿海城市的空气质量普遍好于内陆城市。
B 内陆城市的空气质量普遍好于沿海城市。
C 沿海城市与内陆城市空气质量差不多。
D 暂时无法得出结论。

这里，我们可以进行两样本*t*检验，来查看临海城市与内陆城市的均值差异是否显著。

In [25]:
```
coastal = data[data["Coastal"] == 1]["AQI"]
inland = data[data["Coastal"] == 0]["AQI"]

# 进行方差齐性检验。为后续的两样本t检验服务。
stats.levene(coastal, inland)
```

Out[25]:
```
LeveneResult(statistic=0.08825036641952543, pvalue=0.7666054880248168)
```

In [26]:
```
# 进行两样本t检验。注意，两样本的方差相同与不相同，取得的结果是不同的。
stats.ttest_ind(coastal, inland, equal_var=True)
```

Out[26]:
```
Ttest_indResult(statistic=-2.7303827520948905, pvalue=0.006675422541012958)
```


至此，我们是否可以认为，沿海城市的空气质量普遍好于内陆城市？
A 完全可以。
B 还不可以。
C 有超过99%的几率，可以这样认为。
D 有超过99%的几率，不可以这样认为。

**空气质量主要受哪些因素影响？**

我们很可能会关注某些问题，例如，我们可能会产生类似如下的疑问：

- 人口密度大，是否会对空气质量造成负面影响？
- 绿化率高，是否会提高空气质量？

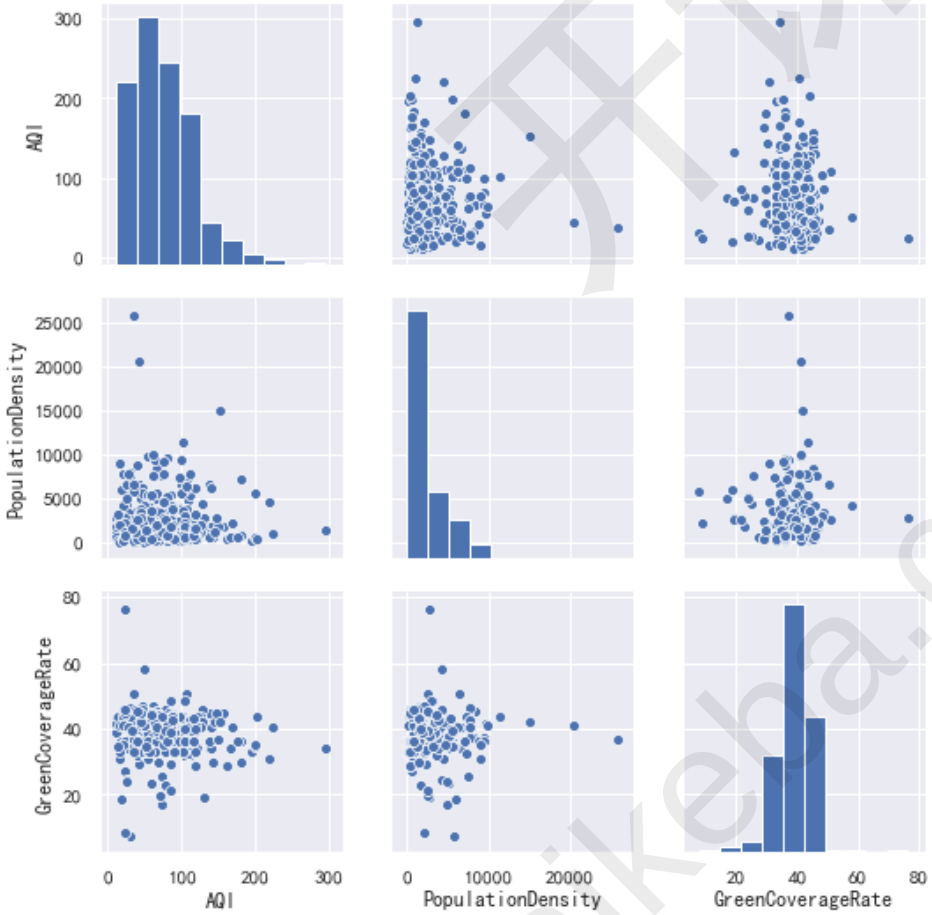


**绘制散点图矩阵**

通过散点图矩阵，可以显示任意两个变量之间的散点图，我们可以通过散点图，观察两个变量之间的关系。

In [27]:
```
sns.pairplot(data[["AQI", "PopulationDensity", "GreenCoverageRate"]])
```

Out[27]: <seaborn.axisgrid.PairGrid at 0xc372a90>




**相关系数**

相关系数，可以用来体现两个连续变量之间的相关性，最为常用的为皮尔逊相关系数。其定义公式为：

$$r(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X)*Var(Y)}}$$

其中，$Cov(X,Y)$为变量$X$与$Y$的协方差，$Var(X)$为$X$的方差，$Var(Y)$为$Y$的方差。

$$Cov(X,Y) = E[(X - E(X))(Y - E(Y))]$$

我们以空气质量（AQI）与降雨量（Precipitation）为例，计算二者的相关系数。

In [28]:
```
x = data["AQI"]
y = data["Precipitation"]
# 计算AQI与Precipitation的协方差。
a = (x - x.mean()) * (y - y.mean())
cov = np.sum(a) / (len(a) - 1)
print(cov)
# 计算AQI与Precipitation的相关系数。
corr = cov / np.sqrt(x.var() * y.var())
print(corr)
```

```
-10098.209013903044
-0.40184407003013883
```

In [29]:
```
print(x.cov(y))
print(x.corr(y))
```

```
-10098.209013903044
-0.40184407003013917
```

In [30]:
```
data.corr()
```

Out[30]:

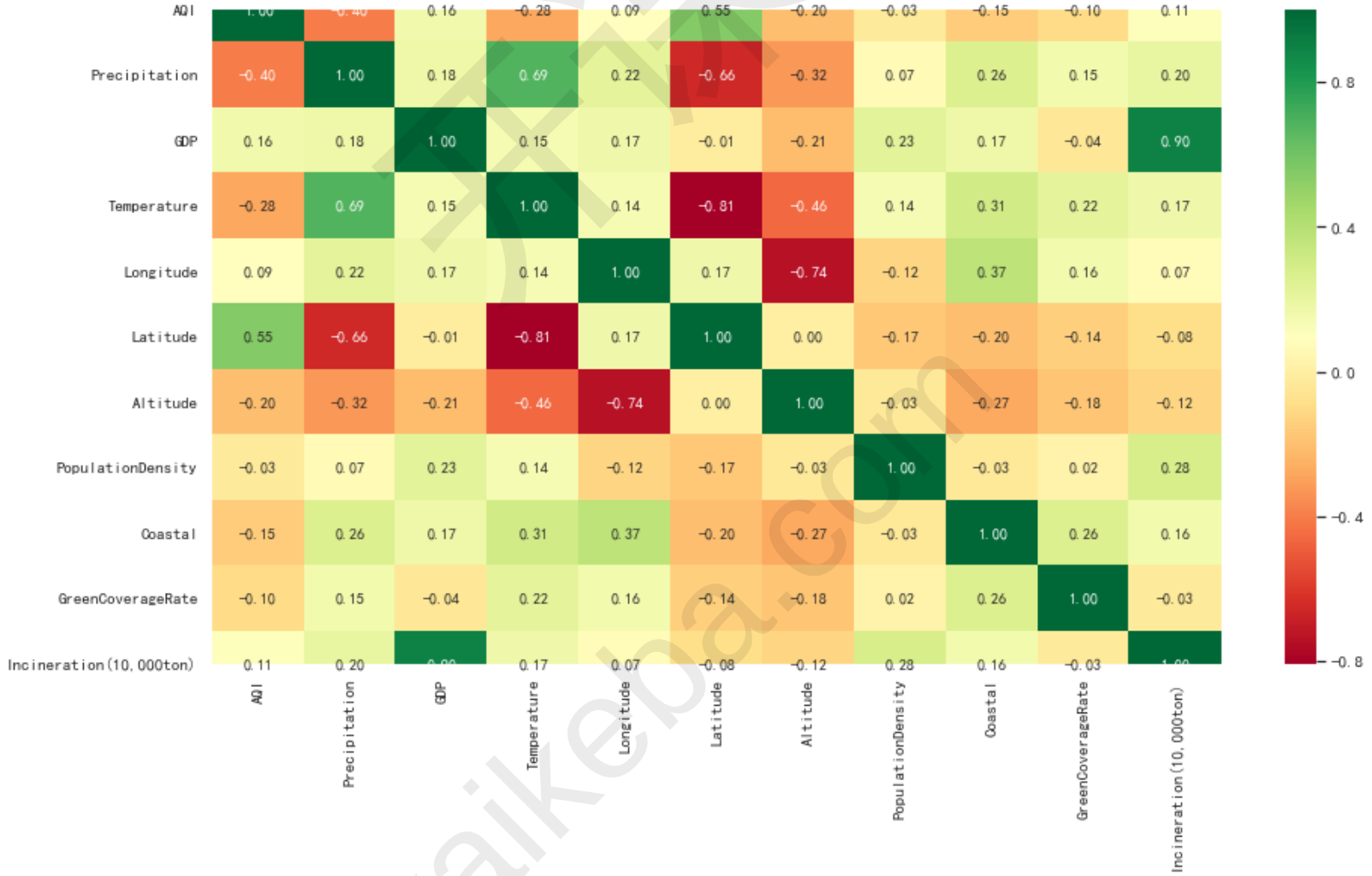
| | AQI | Precipitation | GDP | Temperature | Longitude | Latitude | Altitude | PopulationDensity | Coastal | GreenCoverageRate | Inci |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **AQI** | 1.000000 | -0.401844 | 0.160341 | -0.283956 | 0.093900 | 0.552652 | -0.204753 | -0.026496 | -0.150656 | -0.097734 | |
| **Precipitation** | -0.401844 | 1.000000 | 0.176665 | 0.685447 | 0.223211 | -0.656175 | -0.324124 | 0.067047 | 0.259783 | 0.153291 | |
| **GDP** | 0.160341 | 0.176665 | 1.000000 | 0.145780 | 0.173041 | -0.010124 | -0.208952 | 0.229402 | 0.174241 | -0.039220 | |
| **Temperature** | -0.283956 | 0.685447 | 0.145780 | 1.000000 | 0.141277 | -0.807119 | -0.459426 | 0.144923 | 0.305894 | 0.216575 | |
| **Longitude** | 0.093900 | 0.223211 | 0.173041 | 0.141277 | 1.000000 | 0.173585 | -0.737548 | -0.121986 | 0.374889 | 0.156439 | |
| **Latitude** | 0.552652 | -0.656175 | -0.010124 | -0.807119 | 0.173585 | 1.000000 | 0.002571 | -0.167384 | -0.204199 | -0.142776 | |
| **Altitude** | -0.204753 | -0.324124 | -0.208952 | -0.459426 | -0.737548 | 0.002571 | 1.000000 | -0.031408 | -0.271570 | -0.182449 | |
| **PopulationDensity** | -0.026496 | 0.067047 | 0.229402 | 0.144923 | -0.121986 | -0.167384 | -0.031408 | 1.000000 | -0.034158 | 0.021197 | |
| **Coastal** | -0.150656 | 0.259783 | 0.174241 | 0.305894 | 0.374889 | -0.204199 | -0.271570 | -0.034158 | 1.000000 | 0.264419 | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **GreenCoverageRate** | -0.097734 | 0.153291 | -0.039220 | 0.216575 | 0.156439 | -0.142776 | -0.182449 | 0.021197 | 0.264419 | 1.000000 |
| **Incineration(10,000ton)** | 0.106898 | 0.201174 | 0.899550 | 0.173590 | 0.072068 | -0.081412 | -0.122192 | 0.283563 | 0.158850 | -0.029088 |

为了能够更清晰的呈现相关系数值，我们可以使用热图来展示相关系数。

In [43]:
```
plt.figure(figsize=(15, 8))
sns.heatmap(data.corr(), cmap=plt.cm.RdYlGn, annot=True, fmt=".2f")
```

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x121cf470>



# 课堂练习

观察上图显示的结果，综合来讲，是南方城市空气质量好，还是北方城市空气质量好？
A 南方城市空气好。
B 北方城市空气好。
C 南北方空气质量差不多。
D 无法判断。

**结果统计**

从结果中可知，空气质量指数主要受降雨量（-0.40）与纬度（0.55）影响。

- 降雨量越多，空气质量越好。
- 纬度越低，空气质量越好。

此外，我们还能够发现其他一些明显的细节：

- GDP（城市生产总值）与Incineration（焚烧量）正相关（0.90）。
- Temperature（温度）与Precipitation（降雨量）正相关（0.69）。
- Temperature（温度）与Latitude（纬度）负相关（-0.81）。
- Longitude（经度）与Altitude（海拔）负相关（-0.74）。
- Latitude（纬度）与Precipitation（降雨量）负相关（-0.66）。
- Temperature（温度）与Altitude（海拔）负相关（-0.46）。
- Altitude（海拔）与Precipitation（降雨量）负相关（-0.32）。

**可疑的相关系数值**

通过之前的分析，我们得知，临海城市的空气质量，确实好于内陆城市，可是，为什么临海（Coastal）与空气质量指数（AQI）的相关系数（-0.15）并不高呢？

## 对空气质量指数进行预测

对于某城市，如果我们已知降雨量，温度，经纬度等指标，我们是否能够预测该城市的空气质量指数呢？

答案是肯定的。我们可以通过对以往的数据，去建立一种模式，然后将这种模式去应用于未知的数据，进而预测结果。

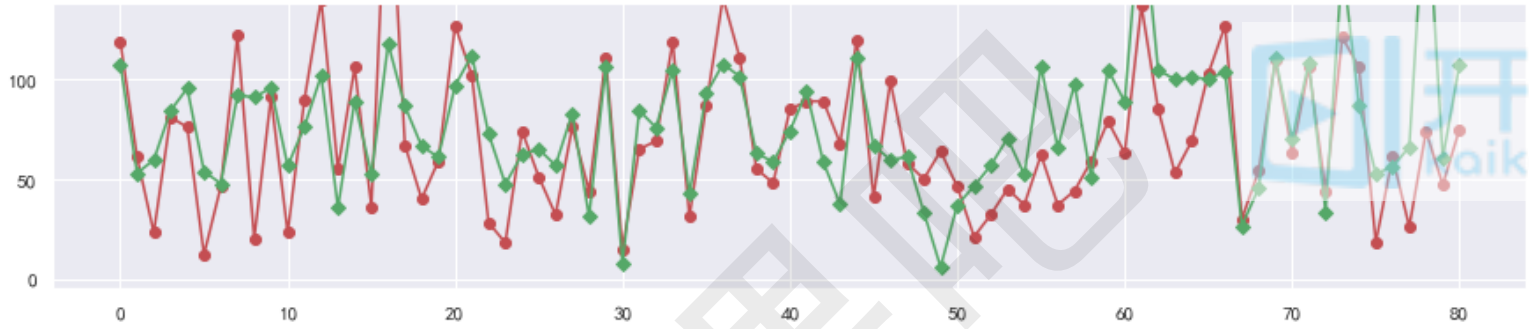
In [37]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

X = data.drop(["City","AQI"], axis=1)
y = data["AQI"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
lr = LinearRegression()
lr.fit(X_train, y_train)
y_hat = lr.predict(X_test)
print(lr.score(X_train, y_train))
print(lr.score(X_test, y_test))
```

```
0.4685357478390665
0.3075998035417721
```

In [38]:
```
plt.figure(figsize=(15, 5))
plt.plot(y_test.values, "-r", label="真实值", marker="o")
plt.plot(y_hat, "-g", label="预测值", marker="D")
plt.legend()
plt.title("线性回归预测结果", fontsize=20)
```

Out[38]: Text(0.5, 1.0, '线性回归预测结果')





之所以线性回归模型拟合的效果不好，是因为数据在高维空间中，并没有呈现线性关系，我们从相关系数中，就可以清楚的看到这点。

In [39]:
```
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n_estimators=500, random_state=0)
rf.fit(X_train, y_train)
y_hat = rf.predict(X_test)
print(rf.score(X_train, y_train))
print(rf.score(X_test, y_test))
```

```
0.9375592254941046
0.6106531491491578
```

In [45]:
```
plt.figure(figsize=(15, 5))
plt.plot(y_test.values, "-r", label="真实值", marker="o")
plt.plot(y_hat, "-g", label="预测值", marker="D")
plt.legend()
plt.title("随机森林预测结果", fontsize=20)
```

Out[45]: Text(0.5, 1.0, '随机森林预测结果')


## 总结

1. 空气质量总体分布上来说，南部城市优于北部城市，西部城市优于东部城市。
2. 临海城市的空气质量整体上好于内陆城市。
3. 是否临海，降雨量与纬度对空气质量指数的影响较大。
4. 我国城市平均空气质量指数大致在(70.63 ~ 80.04)这个区间内，在该区间的可能性概率为95%。
5. 通过历史数据，我们可以对空气质量指数进行预测。