## 10.8 实战：图形探索PM2.5数据

基于前面数据整理实战中的成果，要求：

绘制分年度的PM2.5箱图（所有箱体在一张图上）
分图组绘制PM2.5的直方图
绘制一天24小时PM2.5均值变化的线图
绘制一天24小时PM2.5均值、中位数变化的重叠散点图
各年比较的PM2.5最大值超过100、200、300、500的天数的分段条图
提示：事先需要做较多的数据整理工作

# 11 数据特征的分析探索

## 11.1 数值变量的基本描述

df.describe(

percentiles : 需要输出的百分位数，列表格式提供，如[.25, .5, .75]
include = 'None' : 要求纳入分析的变量类型白名单
None (default) : 只纳入数值变量列
A list-like of dtypes : 列表格式提供希望纳入的类型
'all' : 全部纳入
exclude : 要求剔除出分析的变量类型黑名单，选项同上

)

In [ ]:

```
df2.describe(include = 'all')
```

## 11.2 分类变量的频数统计

Series.value_counts(

normalize = False : 是否返回构成比而不是原始频数
sort = True : 是否按照频数排序(否则按照原始顺序排列)
ascending = False : 是否升序排列
bins : 对数值变量直接进行分段，可看作是pd.cut的简便用法
dropna = True : 结果中是否包括NaN

)

In [ ]:

```
pd.value_counts(df2.类型)
```

In [ ]:

```
pd.value_counts(df2.类型, sort = False)
```

In [ ]:

```
df2.总分.value_counts(bins = 10)
```

## 11.3 交叉表/数据透视表

df.pivot_table(

行列设定
index / columns : 行变量/列变量，多个时以list形式提供
单元格设定
values : 在单元格中需要汇总的变量列，可不写
aggfunc = numpy.mean : 相应的汇总函数
汇总设定
margins = False : 是否加入行列汇总
margins_name = 'All' : 汇总行/列的名称
缺失值处理
fill_value = None : 用于替换缺失值的数值
dropna = True :

)

pd.crosstab(

选项和pivot_table几乎相同
相对而言需要打更多字母，因此使用更麻烦
但是计算频数最方便
输出格式为数据框

行列设定
index / columns : 行变量/列变量，多个时以list形式提供
rownames / colnames = None : 交叉表的行列名称
单元格设定
values : 在单元格中需要汇总的变量列，需要进一步指定aggfunc
aggfunc : 相应的汇总函数
汇总设定
margins = False : 是否加入行列汇总
margins_name = 'All' : 汇总行/列的名称
dropna = True :

)

In [ ]:

```
df2.pivot_table(index = ['所在省份', '主管部门'],
           columns = '类型', values = '总分', aggfunc = sum)
```

In [ ]:

```
pd.crosstab([df2['所在省份'], df2.主管部门],
            df2.类型, values = df2.总分, aggfunc = sum)
```

# 11.4 常用的假设检验方法

相关命令集中在scipy.stats包中，Pandas目前并未考虑做进一步整合，因此仍然需要从Pandas中提取出相应的序列，然后再进行检验

- 更复杂的分析方法可以在statsmodels中实现，而且statsmodels和Pandas高度整合，直接使用Pandas作为其底层数据结构。但对于常用的假设检验方法，使用statsmodels实在是大材小用。

单样本t检验

```
ss.ttest_1samp(a, popmean[, axis])
```

两独立样本t检验

```
ss.ttest_ind(a, b[, axis, equal_var])
```

配对t检验

```
ss.ttest_rel(a, b[, axis])
```

单因素方差分析

```
ss.f_oneway()
```

卡方检验

```
ss.chisquare(f_obs[, f_exp, ddof, axis])
```

相关分析

```
ss.pearsonr(x,y)
```

回归分析

```
ss.linregress(x,y)
```

非参数检验方法

```
kstest(rvs, cdf[, args, N, alternative, mode])
    Perform the Kolmogorov-Smirnov test for goodness of fit.
ks_2samp(data1, data2)
    Computes the Kolmogorov-Smirnov statistic on 2 samples.
rankdata(a[, method])
    Assign ranks to data, dealing with ties appropriately.
mannwhitneyu(x, y[, use_continuity])
    Computes the Mann-Whitney rank test on samples x and y.
tiecorrect(rankvals)
    Tie correction factor for ties in the Mann-Whitney U
    and Kruskal-Wallis H tests.
ranksums(x, y)
    Compute the Wilcoxon rank-sum statistic for two samples.
wilcoxon(x[, y, zero_method, correction])
    Calculate the Wilcoxon signed-rank test.
kruskal(*args)
    Compute the Kruskal-Wallis H-test for independent samples
friedmanchisquare(*args)
    Computes the Friedman test for repeated measurements
```

In [ ]:

```
from scipy import stats as ss
# t 检验
ss.ttest_ind(df2.名次, df2.总分) # 各组分别占一列
```

In [ ]:

```
# ANOVA
ss.f_oneway(df2.名次, df2.总分) # 各组分别占一列
```

In [ ]:

```
# 卡方检验
ss.chisquare(df2.类型.value_counts())
```

In [ ]:

```
# 相关系数
ss.pearsonr(df2.名次, df2.总分)
```

In [ ]:

```
# 简单线性回归
ss.linregress(df2.名次, df2.总分)
```

## 11.5 实战：分析PM2.5数据

基于前面数据整理实战中的成果，要求：

给出分年度的数据基本描述
给出分月份的数据基本描述
按照年月交叉，给出PM2.5的最大值
检验工作日和周末的北京PM2.5数据有无差异

# 12 北京PM2.5变化趋势分析

## 12.1 基本的数据准备

In [ ]:

```
bj
```

In [ ]:

```
# 读入原始数据并建立索引
bj = bj.iloc[:, [1, 6]]
bj.columns = ['datelst', 'value']
bj.set_index(pd.to_datetime(bj.datelst), inplace = True)
bj
```