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
```

In [ ]:

```
# 缺失值处理
bj = bj[bj.value > 0]
bj
```

In [ ]:

```
bj.index.date
```

In [ ]:

```
# 取每日最大值作为当日PM代表
bjana = bj.groupby(bj.index.date).agg(max)
type(bjana.index)
```

In [ ]:

```
# 将索引重建为DatetimeIndex格式
bjana.index = pd.to_datetime(bjana.index)
type(bjana.index)
```

12.2 考察数据的基本分布特征

12.2.1 数据的基本分布

In [ ]:

```
# PM数值的整体分布
bjana.value.plot.hist(bins = 20)
```

In [ ]:

```
# 检查逐月数据缺失情况
pd.crosstab(index=bjana.index.year, columns=bjana.index.month)
```

12.2.2 数据的基本变化规律

In [ ]:

```
bjana.groupby(bjana.index.year).median().plot()
```

In [ ]:

```
bjana.groupby(bjana.index.year).max().plot()
```

In [ ]:

```
bjana.groupby(bjana.index.month).median().plot()
```

In [ ]:

```
bjana.groupby(bjana.index.weekday).median().plot()
```

In [ ]:

```
bj.groupby(bj.index.hour).median().plot()
```

12.3 回答研究问题

12.3.1 对时间周期作必要的调整

In [ ]:

```
# 对年份和月份数值做调整
bjana['month2'] = bjana.index.month
bjana.month2.replace([1, 2], [13, 14], inplace = True)
bjana['year2'] = bjana.index.year
bjana.loc[bjana.month2 > 12, 'year2'] -= 1
bjana['2010']
```

12.3.2 提取秋冬季四个月的数据进行考察

In [ ]:

```
bjana[bjana.month2 > 10].groupby(bjana.year2).value.median().plot()
```

In [ ]:

```
bjana[bjana.month2 > 10].groupby(bjana.year2).value.max().plot()
```

12.3.3 计算爆表天数

In [ ]:

```
bjana[bjana.value > 500].groupby(bjana.year2).value.count().plot.bar()
```

In [ ]:

```
bjana.query("value > 500 and month2 > 10").\
    groupby(bjana.year2).value.count().plot.bar()
```

In [ ]:

```
bjana.query("value > 500 and month2 >= 10").\
    groupby(bjana.year2).value.count().plot.bar()
```