In [1]:

```
import pandas as pd
import numpy as np

# 若想要实现多行输出
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "last"  # 默认为'last'，即输出最后一个结果
```

# 统计分析方法

## 1-描述性统计分析

常用的统计分析指标有计数、求和、求均值、方差、标准差等

- pd对象.describe() 描述性统计为：个数、均值、标准差、
  最小值、25%分位值、50%分位值、75%分位值，以及最大值
- pd对象.value_counts()：统计每一类数据出现的次数
- pd对象.astype('category').describe() → category 类型的描述性统计为：
  非空值个数，类别个数，频次最多的类别，频次最多的次数

In [2]:

```
gra_data = pd.read_excel("../data/grade.xls", sheet_name="grade")
gra_data.head()
```

Out[2]:

| | 学号 | 平时成绩 | 期末成绩 |
|---|---|---|---|
| **0** | 2017001 | 80 | 66 |
| **1** | 2017002 | 87 | 67 |
| **2** | 2017003 | 85 | 60 |
| **3** | 2017004 | 80 | 55 |
| **4** | 2017005 | 76 | 44 |

In [3]:

```
# 新增一列总评成绩：按照平时成绩和期末成绩按3：7的比例获得
gra_data["总评成绩"] = gra_data["平时成绩"] * 0.3 + gra_data["期末成绩"] * 0.7
gra_data.head()
```

Out[3]:

| | 学号 | 平时成绩 | 期末成绩 | 总评成绩 |
|---|---|---|---|---|
| **0** | 2017001 | 80 | 66 | 70.2 |
| **1** | 2017002 | 87 | 67 | 73.0 |
| **2** | 2017003 | 85 | 60 | 67.5 |
| **3** | 2017004 | 80 | 55 | 62.5 |
| **4** | 2017005 | 76 | 44 | 53.6 |

In [4]:

```
gra_data[["平时成绩", "期末成绩"]].describe()
```

Out[4]:

| | 平时成绩 | 期末成绩 |
|---|---|---|
| **count** | 38.000000 | 38.000000 |
| **mean** | 84.657895 | 68.368421 |
| **std** | 6.112797 | 14.695874 |
| **min** | 70.000000 | 32.000000 |
| **25%** | 81.000000 | 60.000000 |
| **50%** | 86.500000 | 68.000000 |
| **75%** | 89.750000 | 81.000000 |
| **max** | 95.000000 | 91.000000 |

## 2-分组分析

df.groupby(by=['分组列1','分组列2'])['统计列1','统计列2'].agg([('聚合名称',聚合函数)])

分组分析是指根据分组字段，将分析对象划分成不同的部分，以对比分析各组之间差异性的分析方法。
分组分析常用的统计指标是计数、求和、平均值

In [5]:

```
df = pd.read_excel("../data/Employee_income.xls", sheet_name="emp_income")
df.head()
```

Out[5]:

| | emp_id | sex | age | education | firstjob | region | industry | occupation | salary | subsidy |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 30 | 男 | 30 | 本科 | 2011-07-20 | 广州 | 机械 | 技术员 | 5000 | 500 |
| **1** | 154 | 男 | 23 | 高中 | 2014-06-23 | 广州 | 机械 | 操作工 | 2500 | 1500 |
| **2** | 40 | 女 | 28 | 大专 | 2011-07-20 | 广州 | 机械 | 文员 | 3800 | 200 |
| **3** | 41 | 女 | 30 | 本科 | 2011-07-20 | 广州 | 机械 | 技术员 | 4500 | 500 |
| **4** | 42 | 男 | 45 | 研究生 | 2000-09-22 | 广州 | 机械 | 主管 | 7699 | 1000 |

## 分组

In [6]:

```
df.groupby(by="education")
```

Out[6]:

<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000002495302CC10>

In [7]:

```
# 选取任意数据块
dict(list(df.groupby(by="education")))["本科"]
```

Out[7]:

| | emp_id | sex | age | education | firstjob | region | industry | occupation | salary | subsidy |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 30 | 男 | 30 | 本科 | 2011-07-20 | 广州 | 机械 | 技术员 | 5000 | 500 |
| **3** | 41 | 女 | 30 | 本科 | 2011-07-20 | 广州 | 机械 | 技术员 | 4500 | 500 |
| **8** | 156 | 男 | 36 | 本科 | 2007-09-22 | 长沙 | 机械 | 技术员 | 4500 | 500 |
| **15** | 154 | 男 | 36 | 本科 | 2011-07-20 | 长沙 | 商业 | 主管 | 4500 | 1000 |
| **17** | 47 | 女 | 36 | 本科 | 2003-03-24 | 广州 | 商业 | 主管 | 5000 | 1500 |

## 聚合

In [8]:

```
#简单聚合
df.groupby(by="education")["salary"].agg([np.mean])
```

Out[8]:

| education | mean |
|---|---|
| 大专 | 3700.000000 |
| 本科 | 4700.000000 |
| 研究生 | 7099.500000 |
| 高中 | 2685.714286 |

In [9]:

```
#多元聚合
df.groupby(by="education")["salary"].agg([np.mean, np.std, np.max, np.min])
```

Out[9]:

| education | mean | std | amax | amin |
|---|---|---|---|---|
| 大专 | 3700.000000 | 627.162924 | 4500 | 3000 |
| 本科 | 4700.000000 | 273.861279 | 5000 | 4500 |
| 研究生 | 7099.500000 | 847.821031 | 7699 | 6500 |
| 高中 | 2685.714286 | 474.090608 | 3500 | 2000 |

In [10]:

```
# 自定义聚合后的列名
df.groupby(by="education")["salary"].agg([("薪资均值", np.mean)])
```

Out[10]:

| education | 薪资均值 |
|---|---|
| 大专 | 3700.000000 |
| 本科 | 4700.000000 |
| 研究生 | 7099.500000 |
| 高中 | 2685.714286 |

In [11]:

```
# 分组聚合
def diff_value(arr):
    return arr.max() - arr.min()


age_result = df.groupby(by=["education", "age"])["salary"].agg(
                        [
                         ("薪资人数", np.size),
                         ("薪资平均值", np.mean),
                         ("薪资最高值", np.max),
                         ("薪资最低值", np.min),
                         ("薪资差值", diff_value)
                        ]
)
age_result
```

Out[11]:

| | | 薪资人数 | 薪资平均值 | 薪资最高值 | 薪资最低值 | 薪资差值 |
|---|---|---|---|---|---|---|
| education | age | | | | | |
| 大专 | 28 | 2 | 3400.000000 | 3800 | 3000 | 800 |
| | 36 | 2 | 4000.000000 | 4500 | 3500 | 1000 |
| 本科 | 30 | 2 | 4750.000000 | 5000 | 4500 | 500 |
| | 36 | 3 | 4666.666667 | 5000 | 4500 | 500 |
| 研究生 | 38 | 1 | 6500.000000 | 6500 | 6500 | 0 |
| | 45 | 1 | 7699.000000 | 7699 | 7699 | 0 |
| 高中 | 23 | 2 | 2250.000000 | 2500 | 2000 | 500 |
| | 25 | 1 | 2500.000000 | 2500 | 2500 | 0 |
| | 28 | 1 | 3000.000000 | 3000 | 3000 | 0 |
| | 37 | 3 | 2933.333333 | 3500 | 2500 | 1000 |

## 3-分布分析

分布分析是指根据分析的目的，将定量数据(**连续型数据**)进行等距或者不等距的分组；
从而研究各组分布规律的一种分析方法，如学生成绩分布、用户年龄分布、收入状况分布等。

在分布分析时，首先用cut()函数确定分布分析中的分层，然后再用groupby()函数实现分组分析

**pd.cut(data,bins,right=true,labels=None,include_lowest=False)**

- data: 进行划分的一维数组
- bins：取整数值，表示将x划分为多少个等距的区间。取序列值，表示将x划分在指定序列中
- right：分组时是否包含右端点，默认为True（包含）
- labels：分组时是否用自定义标签来代替返回的bins，可选项，默认为NULL。
- include_lowest：分组时是否包含左端点，默认为False（不包含）。

按年龄分布状况[20,30,40,50,60]，分组统计人数、平均月薪、最高月薪和最低月薪

In [12]:

```
df = pd.read_excel("../data/Employee_income.xls", sheet_name="emp_income")
df.head(3)
```

Out[12]:

| | emp_id | sex | age | education | firstjob | region | industry | occupation | salary | subsidy |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 30 | 男 | 30 | 本科 | 2011-07-20 | 广州 | 机械 | 技术员 | 5000 | 500 |
| **1** | 154 | 男 | 23 | 高中 | 2014-06-23 | 广州 | 机械 | 操作工 | 2500 | 1500 |
| **2** | 40 | 女 | 28 | 大专 | 2011-07-20 | 广州 | 机械 | 文员 | 3800 | 200 |

In [13]:

```
# 年龄分布状况
age_bins = [20, 30, 40, 50]
age_labels = ["青年", "中年", "中老年"]
df["年龄分层"] = pd.cut(df["age"], age_bins, labels=age_labels,right=False)
df
```

Out[13]:

| | emp_id | sex | age | education | firstjob | region | industry | occupation | salary | subsidy | 年龄分层 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 30 | 男 | 30 | 本科 | 2011-07-20 | 广州 | 机械 | 技术员 | 5000 | 500 | 中年 |
| **1** | 154 | 男 | 23 | 高中 | 2014-06-23 | 广州 | 机械 | 操作工 | 2500 | 1500 | 青年 |
| **2** | 40 | 女 | 28 | 大专 | 2011-07-20 | 广州 | 机械 | 文员 | 3800 | 200 | 青年 |
| **3** | 41 | 女 | 30 | 本科 | 2011-07-20 | 广州 | 机械 | 技术员 | 4500 | 500 | 中年 |
| **4** | 42 | 男 | 45 | 研究生 | 2000-09-22 | 广州 | 机械 | 主管 | 7699 | 1000 | 中老年 |
| **5** | 43 | 男 | 37 | 高中 | 2000-03-23 | 广州 | 商业 | 店员 | 3500 | 600 | 中年 |
| **6** | 44 | 女 | 36 | 大专 | 2003-03-24 | 广州 | 商业 | 主管 | 4500 | 1000 | 中年 |
| **7** | 165 | 女 | 25 | 高中 | 2012-07-21 | 长沙 | 机械 | 文员 | 2500 | 500 | 青年 |
| **8** | 156 | 男 | 36 | 本科 | 2007-09-22 | 长沙 | 机械 | 技术员 | 4500 | 500 | 中年 |
| **9** | 154 | 男 | 38 | 研究生 | 2004-08-12 | 长沙 | 机械 | 主管 | 6500 | 1000 | 中年 |
| **10** | 30 | 男 | 23 | 高中 | 2013-06-23 | 长沙 | 机械 | 操作工 | 2000 | 1000 | 青年 |
| **11** | 154 | 男 | 28 | 大专 | 2011-07-20 | 长沙 | 机械 | 文员 | 3000 | 300 | 青年 |
| **12** | 40 | 女 | 37 | 高中 | 2000-03-23 | 长沙 | 商业 | 店员 | 2500 | 500 | 中年 |
| **13** | 165 | 女 | 36 | 大专 | 2003-03-24 | 长沙 | 商业 | 主管 | 3500 | 1000 | 中年 |
| **14** | 156 | 男 | 37 | 高中 | 2013-06-23 | 长沙 | 商业 | 店员 | 2800 | 1000 | 中年 |
| **15** | 154 | 男 | 36 | 本科 | 2011-07-20 | 长沙 | 商业 | 主管 | 4500 | 1000 | 中年 |
| **16** | 46 | 女 | 28 | 高中 | 2007-03-23 | 广州 | 商业 | 店员 | 3000 | 1500 | 青年 |
| **17** | 47 | 女 | 36 | 本科 | 2003-03-24 | 广州 | 商业 | 主管 | 5000 | 1500 | 中年 |

In [14]:

```
# 分组统计人数、平均月薪、最高月薪和最低月薪
aggResult = df.groupby(by=["年龄分层"])["salary"].agg(
                    [("人数", np.size)
                     , ("平均月薪", np.mean)
                     , ("最高月薪", np.max)
                     , ("最低月薪", np.min)
                    ]
)
aggResult
```

Out[14]:

| 年龄分层 | 人数 | 平均月薪 | 最高月薪 | 最低月薪 |
|---|---|---|---|---|
| 青年 | 6 | 2800.000000 | 3800 | 2000 |
| 中年 | 11 | 4254.545455 | 6500 | 2500 |
| 中老年 | 1 | 7699.000000 | 7699 | 7699 |

## 4-交叉分析

交叉分析通常用于分析两个或两个以上分组变量之间的关系，以交叉表形式进行变量间关系的对比分析；
从数据的不同维度，综合进行分组细分，进一步了解数据的构成、分布特征。交叉分析有数据透视表和交叉表两种：

### 1. 透视表

pivot_table() 函数返回值是数据透视表的结果，该函数相当于Excel中的数据透视表功能。

**透视表** pd.pivot_table(data,values,index,columns,agg,func,fill_value,margins)

- data： 要应用透视表的数据框。
- values： 待聚合的列名，默认聚合所有数值列。
- index： 用于分组的列名或其他分组键，出现在结果透视表的行。
- columns： 用于分组的列名或其他分组键，出现在结果透视表的列。
- aggfunc： 聚合函数或函数列表，默认为'mean'，可以是任何对groupby有效的函数。
- fill_value： 用于替换结果表中的缺失值。
- margins： 添加行/列小计和总计，默认为False。

**在交叉分析前，先用cut()函数确定交叉分析中的分层，然后再利用pivot_table()函数实现交叉分析。**

In [15]:

```
df = pd.read_excel("../data/Employee_income.xls", sheet_name="emp_income")
df.head(3)
```

Out[15]:

| | emp_id | sex | age | education | firstjob | region | industry | occupation | salary | subsidy |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 30 | 男 | 30 | 本科 | 2011-07-20 | 广州 | 机械 | 技术员 | 5000 | 500 |
| **1** | 154 | 男 | 23 | 高中 | 2014-06-23 | 广州 | 机械 | 操作工 | 2500 | 1500 |
| **2** | 40 | 女 | 28 | 大专 | 2011-07-20 | 广州 | 机械 | 文员 | 3800 | 200 |

In [16]:

```
age_bins = [20, 30, 40, 50, 60, 70]
age_labels = ["20-29岁", "30-39岁", "40-49岁", "50-59岁", "60-69岁"]
df["年龄分层"] = pd.cut(df.age, age_bins, right=False, labels=age_labels)
df.head(3)
```

Out[16]:

| | emp_id | sex | age | education | firstjob | region | industry | occupation | salary | subsidy | 年龄分层 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 30 | 男 | 30 | 本科 | 2011-07-20 | 广州 | 机械 | 技术员 | 5000 | 500 | 30-39岁 |
| **1** | 154 | 男 | 23 | 高中 | 2014-06-23 | 广州 | 机械 | 操作工 | 2500 | 1500 | 20-29岁 |
| **2** | 40 | 女 | 28 | 大专 | 2011-07-20 | 广州 | 机械 | 文员 | 3800 | 200 | 20-29岁 |

***(1)对年龄（age）和性别（sex）数据列进行交叉分析，求解不同年龄段中不同性别的平均年龄（人数）。***

In [17]:

```
ptResult1 = pd.pivot_table(
    df,
    values=["age"],
    index=["年龄分层"],
    columns=["sex"],
    aggfunc=[np.size],
    margins=True,
    # fill_value="无",
)
ptResult1
```

Out[17]:

| | size | | |
|---|---|---|---|
| | age | | |
| sex | 女 | 男 | All |
| 年龄分层 | | | |
| 20-29岁 | 3.0 | 3.0 | 6 |
| 30-39岁 | 5.0 | 6.0 | 11 |
| 40-49岁 | NaN | 1.0 | 1 |
| All | 8.0 | 10.0 | 18 |

***(2) 对年龄 (age) 和学历 (education) 数据列进行交叉分析，求解不同年龄段中不同学历员工的平均月薪。***

data.pivot_table(values,index,columns,agg,func,fill_value,margins)

In [18]:

```
# 分组统计人数、平均月薪、最高月薪和最低月薪
aggResult = df.groupby(by=["年龄分层"])["salary"].agg(
                    {  ("人数", np.size)
                     , ("平均月薪", np.mean)
                     , ("最高月薪", np.max)
                     , ("最低月薪", np.min)
                    }
)
aggResult
```

Out[18]:

| 年龄分层 | 平均月薪 | 最高月薪 | 最低月薪 | 人数 |
|---|---|---|---|---|
| **20-29岁** | 2800.000000 | 3800.0 | 2000.0 | 6.0 |
| **30-39岁** | 4254.545455 | 6500.0 | 2500.0 | 11.0 |
| **40-49岁** | 7699.000000 | 7699.0 | 7699.0 | 1.0 |
| **50-59岁** | NaN | NaN | NaN | NaN |
| **60-69岁** | NaN | NaN | NaN | NaN |

## 5-结构分析

结构分析是在分组和交叉的基础上，计算各组成部分所占的比例，进而分析总体的内部特征的一种分析方法。重点在于了解各部分占总体的比例,例如:求公司中不同学历员工所占的比例，产品在市场的占有率、股权结构等

在结构分析时，先利用pivot_table()函数进行数据透视表分析，
然后，指定axis参数对数据透视表按行或列进行计算聚合函数(add,sub,multiply,div;sum,mean,var,sd)
（**当axis=0时按列计算，axis=1时按行计算**）。

**对年龄（age）和学历（education）数据列进行结构分析，不同年龄分层下各种学历的占比**

In [19]:

```
# 年龄分布状况
age_bins = [20, 30, 40, 50, 60, 70]
age_labels = ["20-29岁", "30-39岁", "40-49岁", "50-59岁", "60-69岁"]
df["年龄分层"] = pd.cut(df.age, age_bins, right=False, labels=age_labels)
#透视表
ptResult = pd.pivot_table(df,
                          values=["age"],
                          index=["年龄分层"],
                          columns=["education"],
                          aggfunc=[np.size]
                         ,fill_value=0
)
ptResult
```

Out[19]:

| | size | | | |
|---|---|---|---|---|
| | age | | | |
| education | 大专 | 本科 | 研究生 | 高中 |
| 年龄分层 | | | | |
| 20-29岁 | 2 | 0 | 0 | 4 |
| 30-39岁 | 2 | 5 | 1 | 3 |
| 40-49岁 | 0 | 0 | 1 | 0 |

In [20]:

```
ptResult.sum(axis=0)  # 对每一列求和
```

Out[20]:

```
              education
size  age     大专          4
              本科          5
              研究生         2
              高中          7
dtype: int64
```

In [21]:

```
# 按行求占比(每一行特征除以除以每一列的总和)
a = ptResult.div(ptResult.sum(axis=0), axis=1)
a
# div的第一个参数是除法的分子，意思是按行把数据除以该列的总和。即得到某一个年龄分层下，学历占比。
```

Out[21]:

| | size | | | |
|---|---|---|---|---|
| | age | | | |
| education | 大专 | 本科 | 研究生 | 高中 |
| 年龄分层 | | | | |
| 20-29岁 | 0.5 | 0.0 | 0.0 | 0.571429 |
| 30-39岁 | 0.5 | 1.0 | 0.5 | 0.428571 |
| 40-49岁 | 0.0 | 0.0 | 0.5 | 0.000000 |

In [22]:

```
# 按列求占比(每一列样本除以除以每一行的总和)
ptResult.div(ptResult.sum(axis=1), axis=0)
```

Out[22]:

| | size | | | |
|---|---|---|---|---|
| | age | | | |
| education | 大专 | 本科 | 研究生 | 高中 |
| 年龄分层 | | | | |
| 20-29岁 | 0.333333 | 0.000000 | 0.000000 | 0.666667 |
| 30-39岁 | 0.181818 | 0.454545 | 0.090909 | 0.272727 |
| 40-49岁 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |

In [23]:

```
ptResult.sum(axis=1)  # 对每一行求和
```

Out[23]:

```
年龄分层
20-29岁     6
30-39岁    11
40-49岁     1
dtype: int64
```

## 6-相关分析

相关分析（Correlation Analysis）用于研究现象之间是否存在某种依存关系,
并探讨具有依存关系的现象的相关方向以及相关程度,是研究随机变量之间相关关系的一种统计方法。
线性相关关系主要采用皮尔逊（Pearson）相关系数r来度量连续变量之间线性相关强度:

r>0，线性正相关；
r<0，线性负相关；

| 相关系数\|r\|取值范围 | 相关程度 |
| --- | --- |
| 0⩽\|r\|<0.3 | 低度相关 |
| 0.3⩽\|r\|<0.8 | 中度相关 |
| 0.8⩽\|r\|⩽1 | 高度相关 |

**相关分析函数包括DataFrame.corr()和Series.corr(other):**
**计算相关系数的corr()函数只会对数据框中的数据列进行计算**

- 如果由数据框调用corr()函数，那么将会计算列与列之间的相似度。
- 如果由序列调用corr()方法，那么只是该序列与传入的序列之间的相关度。函数返回值如下。
- DataFrame调用：返回DataFrame。
- Series调用：返回一个数值型数据，大小为相关度。

In [24]:

```
# 计算age和salary的相关系数
corrResult1 = df["age"].corr(df["salary"])
corrResult1
```

Out[24]:

0.6781676305144909

In [25]:

```
# 计算age和salary、subsidy的相关系数
corrResult2 = df[["age", "salary", "subsidy"]].corr()
corrResult2
```

Out[25]:

| | age | salary | subsidy |
| --- | --- | --- | --- |
| **age** | 1.000000 | 0.678168 | 0.062137 |
| **salary** | 0.678168 | 1.000000 | 0.067629 |
| **subsidy** | 0.062137 | 0.067629 | 1.000000 |

In [26]:

```
# 返回一个相关系数矩阵
df.corr()
```

Out[26]:

| | emp_id | age | salary | subsidy |
|---|---|---|---|---|
| **emp_id** | 1.000000 | -0.029849 | -0.143666 | 0.043537 |
| **age** | -0.029849 | 1.000000 | 0.678168 | 0.062137 |
| **salary** | -0.143666 | 0.678168 | 1.000000 | 0.067629 |
| **subsidy** | 0.043537 | 0.062137 | 0.067629 | 1.000000 |

## 小结:

1. 描述性统计分析

- pd对象.describe() 描述性统计为：个数、均值、标准差、
  最小值、25%分位值、50%分位值、75%分位值，以及最大值
- pd对象.value_counts()：统计每一类数据出现的次数

2. 分组分析（类别型离散数据分组分析）

- data.groupby(by=['分组列1','分组列2'])[['统计列1','统计列2']].agg({

  ('聚合列名',聚合函数)
  })

3. 分布分析(连续型数据分组分析):
   先用cut()函数确定分布分析中的分层，再用groupby()分组分析得出结果

- pd.cut(data,bins,labels)

4. 交叉分析（透视表）

- ptResult = pd.pivot_table( data #data:完整的DataFram

```
  ,values=['金额']       #values：待统计列的列名
  ,index=['类别']        #index:交叉表行索引取值所在列名
  ,columns=['月份']      #columns:交叉表列索引取值所在列名
  ,aggfunc=[np.sum]      #aggfunc：统计方法的函数名
  , margins=True         #margins：是否添加行/列小计和总计
  ,fill_value='无'       #fill_value：缺失值显示方式
)
```

5. 结构分析:
   先利用pivot_table()函数进行数据透视表分析;
   指定axis参数对数据透视表按行或列进行计算（axis=0：按列计算，axis=1：按行计算）

- 按列求占比(元素在列中的占比)：ptResult.div(ptResult.sum(axis=0), axis=1)
- 按行求占比(元素在行中的占比)：ptResult.div(ptResult.sum(axis=1), axis=0)

6. 相关分析：(系数矩阵中值越大，相关性越强，相关系数为1，则说明两列值相同)

- dataFrame[['列名1', '列名2','列名3']].corr()：返回多列之间的相关系数矩阵

In [ ]: