知乎 首发于 Data Analyst

特征工程（二）数据分析的六基本思路

Alan
数据分析、挖掘、机器学习

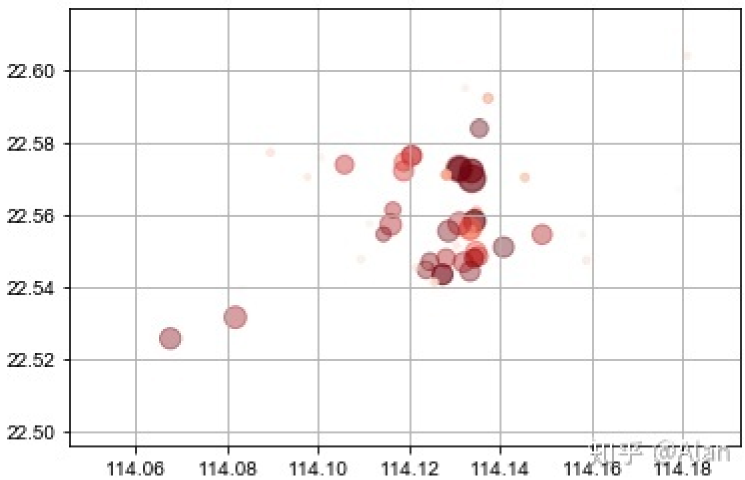
【目录】

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
#显示所有字体格式，解决plt画图，标签中文乱码
from matplotlib.font_manager import FontManager
fm = FontManager()
mat_fonts = set(f.name for f in fm.ttflist)
plt.rcParams['font.sans-serif'] = ['Arial Unicode MS']
```

1、分布分析

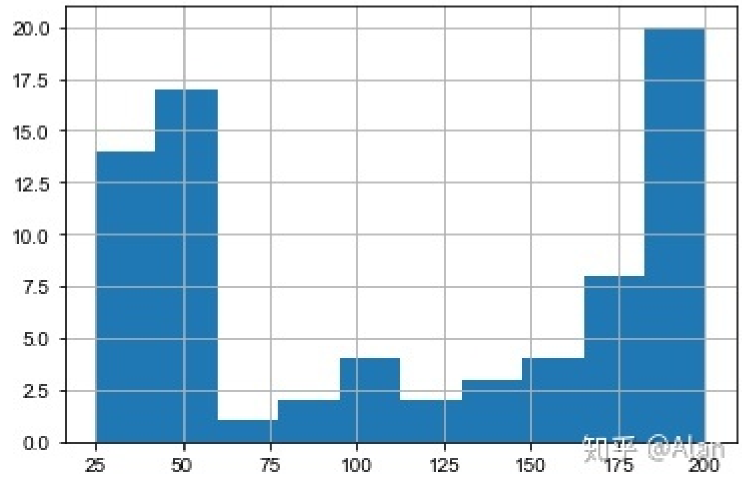
```
data = pd.read_csv('/Users/ouminyang/Downloads/second_hand_ house.csv')
plt.scatter(data['经度'],data['纬度'],  # 按照经纬度显示
            s = data['房屋单价']/500,  # 按照单价显示大小
            c = data['参考总价'],  # 按照总价显示颜色
            alpha = 0.4, cmap = 'Reds')
plt.grid()
print(data.dtypes)
print('-------\n数据长度为%i条' % len(data))
data.head()
# 通过数据可见，一共8个字段
# 定量字段：房屋单价，参考首付，参考总价，*经度，*纬度，*房屋编码
# 定性字段：小区，朝向
房屋编码      int64
小区        object
朝向        object
房屋单价      int64
参考首付    float64
参考总价    float64
经度      float64
纬度      float64
dtype: object
-------
数据长度为75条
```

0	605093949	大望新平村	南北	5434	15.0	50.0	114.180964	22.603698
1	605768856	通宝楼	南北	3472	7.5	25.0	114.179298	22.566910
2	606815561	罗湖区罗芳村	南北	5842	15.6	52.0	114.158869	22.547223
3	605147285	兴华苑	南北	3829	10.8	36.0	114.158040	22.554343
4	606030866	京基东方都会	西南	47222	51.0	170.0	114.149243	22.564376




```
# 极差：max-min
# 只针对定量字段

def d_range(df,*cols):
    krange = []
    for col in cols:
        crange = df[col].max() - df[col].min()
        krange.append(crange)
    return(krange)
# 创建函数求极差

key1 = '参考首付'
key2 = '参考总价'
dr = d_range(data,key1,key2)
print('%s极差为 %f \n%s极差为 %f' % (key1, dr[0], key2, dr[1]))
# 求出数据对应列的极差
参考首付极差为 52.500000
参考总价极差为 175.000000
# 频率分布情况 - 定量字段
# ① 通过直方图直接判断分组组数
```

```
# 简单查看数据分布，确定分布组数 → 一般8-16即可
# 这里以10组为参考
<matplotlib.axes._subplots.AxesSubplot at 0x11ba9fd50>
```




```
# 频率分布情况 - 定量字段
# ② 求出分组区间

gcut = pd.cut(data[key2],10,right=False)#分组区间的
gcut_count = gcut.value_counts(sort=False)  # 不排序
data['%s分组区间' % key2] = gcut.values
print(gcut.head(),'\n------')
print(gcut_count)
data.head()
# pd.cut(x, bins, right)：按照组数对x分组，且返回一个和x同样长度的分组dataframe，right → 是
# 通过groupby查看不同组的数据频率分布
# 给源数据data添加“分组区间”列
0      [42.5, 60.0)
1      [25.0, 42.5)
2      [42.5, 60.0)
3      [25.0, 42.5)
4    [165.0, 182.5)
Name: 参考总价, dtype: category
Categories (10, interval[float64]): [[25.0, 42.5) < [42.5, 60.0) < [60.0, 77.5) < [77.
------
[25.0, 42.5)       14
[42.5, 60.0)       17
[60.0, 77.5)        1
[77.5, 95.0)        2
[95.0, 112.5)       4
[112.5, 130.0)      2
```

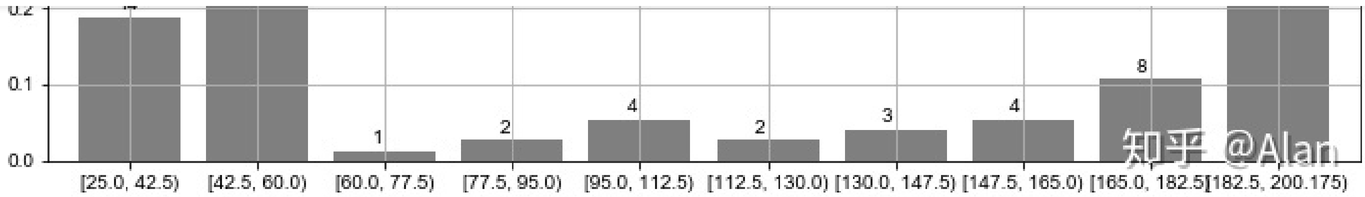
```
[165.0, 182.5)       8
[182.5, 200.175)    20
Name: 参考总价, dtype: int64
```

	房屋编码	小区	朝向	房屋单价	参考首付	参考总价	经度	纬度	参考总价分组区间
0	605093949	大望新平村	南北	5434	15.0	50.0	114.180964	22.603698	[42.5, 60.0)
1	605768856	通宝楼	南北	3472	7.5	25.0	114.179298	22.566910	[25.0, 42.5)
2	606815561	罗湖区罗芳村	南北	5842	15.6	52.0	114.158869	22.547223	[42.5, 60.0)
3	605147285	兴华苑	南北	3829	10.8	36.0	114.158040	22.554343	[25.0, 42.5)
4	606030866	京基东方都会	西南	47222	51.0	170.0	114.149243	22.554370	[165.0, 182.5)

```
# 频率分布情况 - 定量字段
# ③  求出目标字段下频率分布的其他统计量 → 频数，频率，累计频率

r_zj = pd.DataFrame(gcut_count)
r_zj.rename(columns ={gcut_count.name:'频数'}, inplace = True)  # 修改频数字段名
r_zj['频率'] = r_zj / r_zj['频数'].sum()  # 计算频率
r_zj['累计频率'] = r_zj['频率'].cumsum()  # 计算累计频率
r_zj['频率%'] = r_zj['频率'].apply(lambda x: "%.2f%%" % (x*100))  # 以百分比显示频率
r_zj['累计频率%'] = r_zj['累计频率'].apply(lambda x: "%.2f%%" % (x*100))  # 以百分比显示
r_zj.style.bar(subset=['频率','累计频率'], color='green',width=100)
# 可视化显示
```

[25.0, 42.5)	14	0.186667	0.186667	18.67%	18.67%
[42.5, 60.0)	17	0.226667	0.413333	22.67%	41.33%
[60.0, 77.5)	1	0.0133333	0.426667	1.33%	42.67%
[77.5, 95.0)	2	0.0266667	0.453333	2.67%	45.33%
[95.0, 112.5)	4	0.0533333	0.506667	5.33%	50.67%
[112.5, 130.0)	2	0.0266667	0.533333	2.67%	53.33%
[130.0, 147.5)	3	0.04	0.573333	4.00%	57.33%
[147.5, 165.0)	4	0.0533333	0.626667	5.33%	62.67%
[165.0, 182.5)	8	0.106667	0.733333	10.67%	73.33%
[182.5, 200.175)	20	0.266667	1	26.67%	100.00%

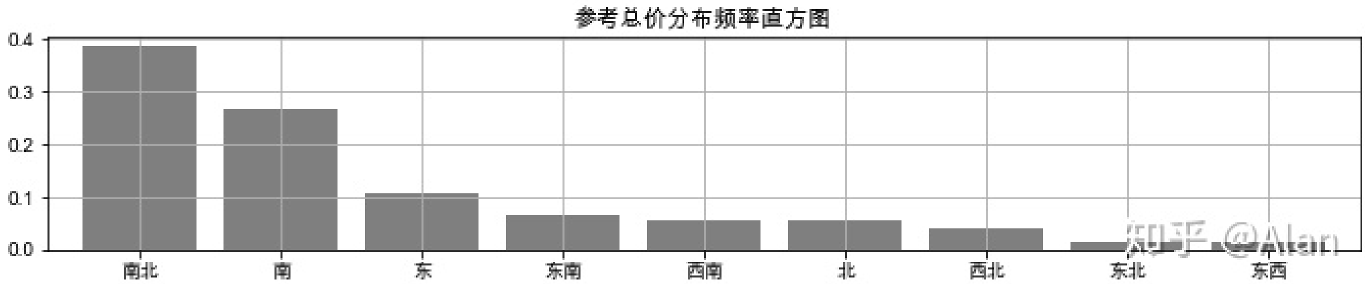

```
# 频率分布情况 - 定量字段
# ④ 绘制频率直方图

r_zj['频率'].plot(kind = 'bar',
                 width = 0.8,
                 figsize = (12,2),
                 rot = 0,
                 color = 'k',
                 grid = True,
                 alpha = 0.5)
plt.title('参考总价分布频率直方图')
# 绘制直方图

x = len(r_zj)
y = r_zj['频率']
m = r_zj['频数']
for i,j,k in zip(range(x),y,m):
    plt.text(i-0.1,j+0.01,'%i' % k, color = 'k')
# 添加频数标签
```




```
# 频率分布情况 - 定性字段
# ① 通过计数统计判断不同类别的频率

cx_g = data['朝向'].value_counts(sort=True)
print(cx_g)
# 统计频率

r_cx = pd.DataFrame(cx_g)
r_cx.rename(columns ={cx_g.name:'频数'}, inplace = True)  # 修改频数字段名
r_cx['频率'] = r_cx / r_cx['频数'].sum()  # 计算频率
r_cx['累计频率'] = r_cx['频率'].cumsum()  # 计算累计频率
r_cx['频率%'] = r_cx['频率'].apply(lambda x: "%.2f%%" % (x*100))  # 以百分比显示频率
r_cx['累计频率%'] = r_cx['累计频率'].apply(lambda x: "%.2f%%" % (x*100))  # 以百分比显示
r_cx.style.bar(subset=['频率','累计频率'], color='#d65f5f',width=100)
# 可视化显示
南北    29
南     20
东      8
东南     5
西南     4
北      4
西北     3
东北     1
东西     1
Name: 朝向, dtype: int64
```

南北	29	0.386667	0.386667	38.67%	38.67%
南	20	0.266667	0.653333	26.67%	65.33%
东	8	0.106667	0.76	10.67%	76.00%
东南	5	0.0666667	0.826667	6.67%	82.67%
西南	4	0.0533333	0.88	5.33%	88.00%
北	4	0.0533333	0.933333	5.33%	93.33%
西北	3	0.04	0.973333	4.00%	97.33%
东北	1	0.0133333	0.986667	1.33%	98.67%
东西	1	0.0133333	1	1.33%	100.00%


```
# 频率分布情况 - 定量字段
# ② 绘制频率直方图、饼图

plt.figure(num = 1,figsize = (12,2))
r_cx['频率'].plot(kind = 'bar',
                 width = 0.8,
                 rot = 0,
                 color = 'k',
                 grid = True,
                 alpha = 0.5)
plt.title('参考总价分布频率直方图')
# 绘制直方图

plt.figure(num = 2)
plt.pie(r_cx['频数'],
       labels = r_cx.index,
       autopct='%.2f%%',
       shadow = True)
plt.axis('equal')
# 绘制饼图
(-1.1101621526291232,
 1.1004839130571389,
```



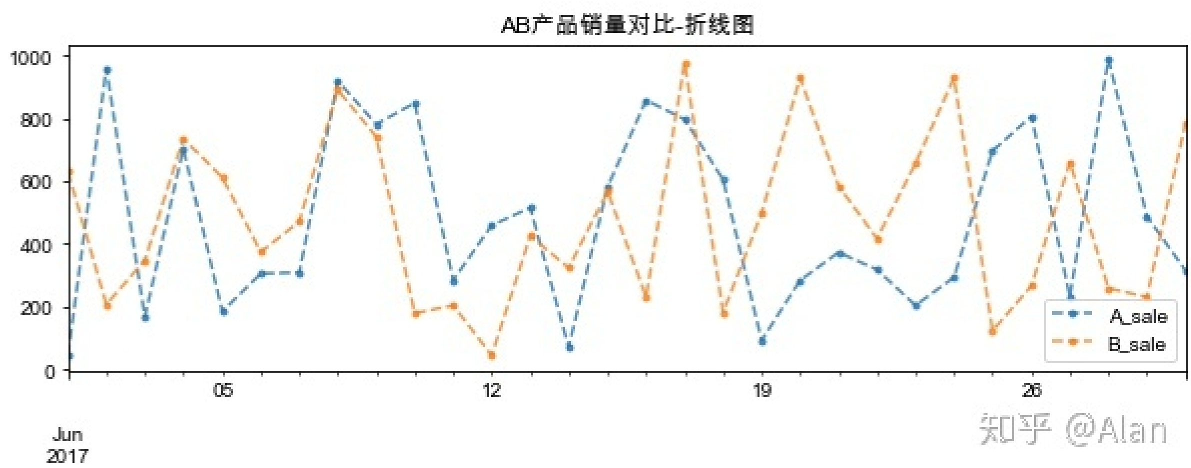
2、对比分析

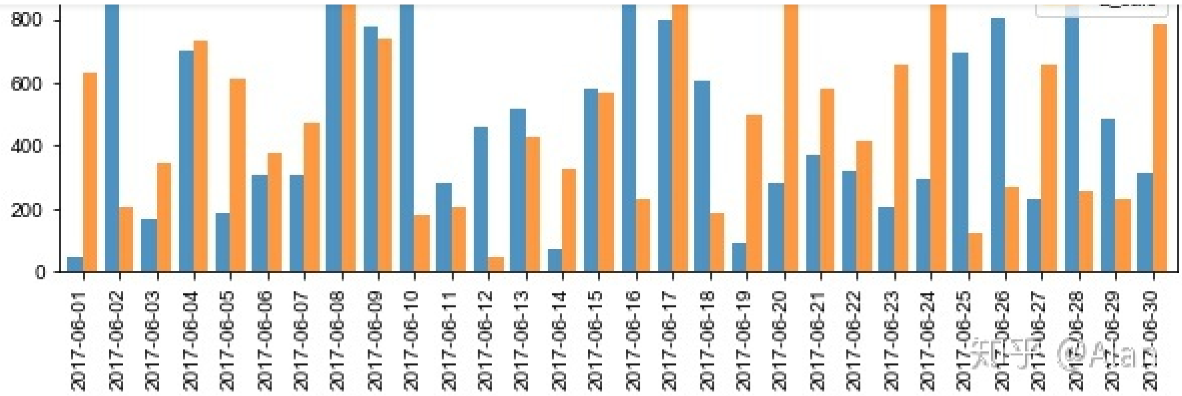
对比分析 → 两个互相联系的指标进行比较

```
1、绝对数比较（相减） / 相对数比较（相除）
2、结构分析、比例分析、空间比较分析、动态对比分析
#1、绝对数比较 → 相减
# 相互对比的指标在量级上不能差别过大
# （1）折线图比较
# （2）多系列柱状图比较

data = pd.DataFrame(np.random.rand(30,2)*1000,
                   columns = ['A_sale','B_sale'],
                   index = pd.period_range('20170601','20170630'))
print(data.head())
# 创建数据 → 30天内A/B产品的日销售额

data.plot(kind='line',
       style = '--.',
       alpha = 0.8,
```

```
# 折线图比较

data.plot(kind = 'bar',
          width = 0.8,
          alpha = 0.8,
          figsize = (10,3),
          title = 'AB产品销量对比-柱状图')
# 多系列柱状图比较
A_sale      B_sale
2017-06-01   45.640267  630.139065
2017-06-02  959.827223  208.128640
2017-06-03  168.072333  347.706965
2017-06-04  703.102723  734.915008
2017-06-05  185.494455  611.327248
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11ca92bd0>
```

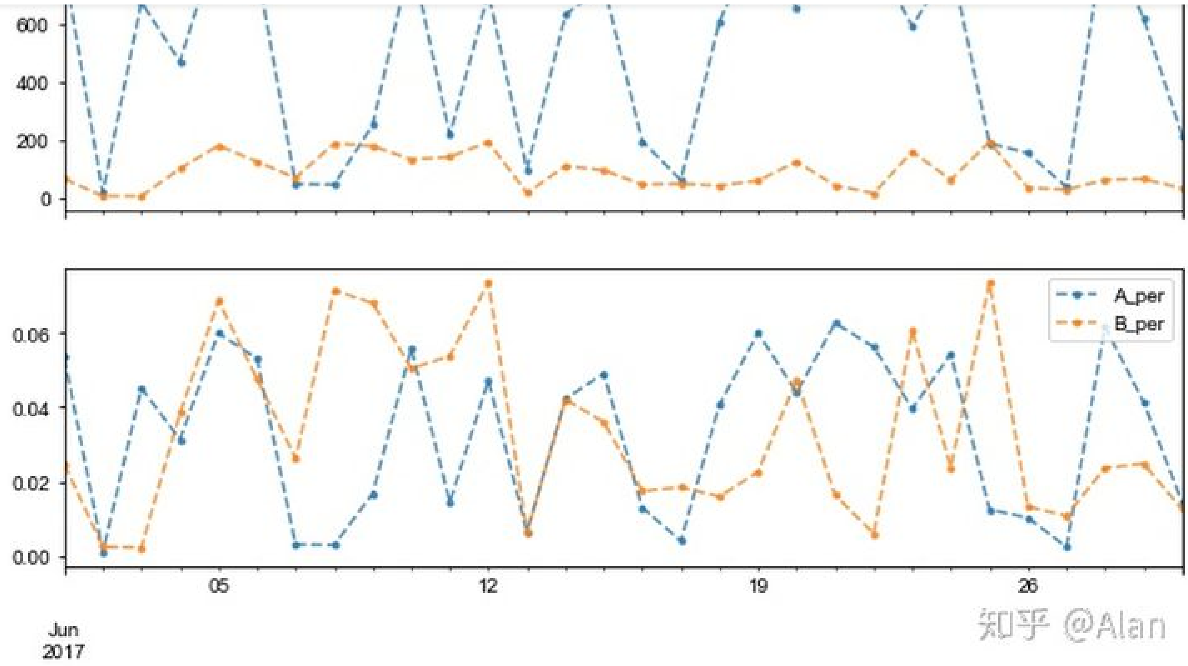




```
# 1、绝对数比较 → 相减
# （3）柱状图堆叠图+差值折线图比较

fig3 = plt.figure(figsize=(10,6))
plt.subplots_adjust(hspace=0.3)
# 创建子图及间隔设置

ax1 = fig3.add_subplot(2,1,1)
x = range(len(data))
y1 = data['A_sale']
y2 = -data['B_sale']
plt.bar(x,y1,width = 1,facecolor = 'yellowgreen')
plt.bar(x,y2,width = 1,facecolor = 'lightskyblue')
plt.title('AB产品销量对比-堆叠图')
plt.grid()
plt.xticks(range(0,30,6))
ax1.set_xticklabels(data.index[::6])
# 创建堆叠图

ax2 = fig3.add_subplot(2,1,2)
y3 = data['A_sale']-data['B_sale']
plt.plot(x,y3,'--go')

plt.grid()
plt.title('AB产品销量对比-差值折线')
plt.xticks(range(0,30,6))
ax2.set_xticklabels(data.index[::6])
# 创建差值折线图
[Text(0, 0, '2017-06-01'),
 Text(0, 0, '2017-06-07'),
 Text(0, 0, '2017-06-13'),
 Text(0, 0, '2017-06-19'),
 Text(0, 0, '2017-06-25')]
```


```
# 2、相对数比较 → 相除
# 有联系的指标综合计算后的对比，数值为相对数
# 结构分析、比例分析、空间比较分析、动态对比分析、计划完成度分析
#  （1）结构分析
# 在分组基础上，各组总量指标与总体的总量指标对比，计算出各组数量在总量中所占比重
# 反映总体的内部结构

data = pd.DataFrame({'A_sale':np.random.rand(30)*1000,
                    'B_sale':np.random.rand(30)*200},
                   index = pd.period_range('20170601','20170630'))
print(data.head())
print('------')
# 创建数据 → 30天内A/B产品的日销售额
# A/B产品销售额量级不同

data['A_per'] = data['A_sale'] / data['A_sale'].sum()
data['B_per'] = data['B_sale'] / data['B_sale'].sum()
# 计算出每天的营收占比

data['A_per%'] = data['A_per'].apply(lambda x: '%.2f%%' % (x*100))
data['B_per%'] = data['B_per'].apply(lambda x: '%.2f%%' % (x*100))
# 转换为百分数
print(data.head())

fig,axes = plt.subplots(2,1,figsize = (10,6),sharex=True)
```

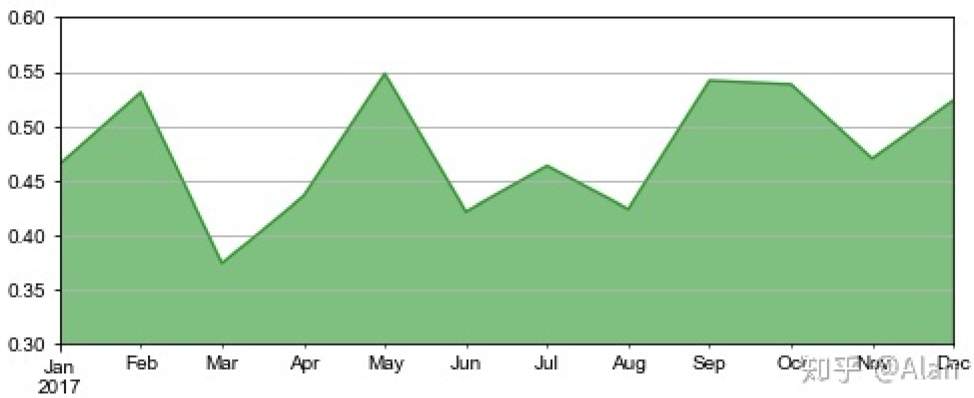
```
data[['A_per','B_per']].plot(kind='line',style = '--.',alpha = 0.8,ax=axes[1])
axes[1].legend(loc = 'upper right')
# 绝对值对比较难看出结构性变化，通过看销售额占比来看售卖情况的对比

# 同时可以反应"强度" → 两个性质不同但有一定联系的总量指标对比，用来说明"强度"、"密度"、"普遍程度
# 例如：国内生产总值"元/人"，人口密度"人/平方公里"
A_sale      B_sale
2017-06-01  796.473461   64.896401
2017-06-02   16.939405    6.695858
2017-06-03  672.849801    6.244486
2017-06-04  468.034645  100.609225
2017-06-05  892.819851  179.664584
------
                A_sale      B_sale     A_per     B_per A_per% B_per%
2017-06-01  796.473461   64.896401  0.053431  0.024771  5.34%  2.48%
2017-06-02   16.939405    6.695858  0.001136  0.002556  0.11%  0.26%
2017-06-03  672.849801    6.244486  0.045138  0.002384  4.51%  0.24%
2017-06-04  468.034645  100.609225  0.031398  0.038403  3.14%  3.84%
2017-06-05  892.819851  179.664584  0.059895  0.068578  5.99%  6.86%




<matplotlib.legend.Legend at 0x11cbe5190>
```




```
# 2、相对数比较 → 相除
# （2）比例分析
# 在分组的基础上，将总体不同部分的指标数值进行对比，其相对指标一般称为"比例相对数"
# 比例相对数 = 总体中某一部分数值 / 总体中另一部分数值 → "基本建设投资额中工业、农业、教育投资的

data = pd.DataFrame({'consumption':np.random.rand(12)*1000 + 2000,
                     'salary':np.random.rand(12)*500 + 5000},
                    index = pd.period_range('2017/1','2017/12',freq = 'M'))
print(data.head())
print('------')
# 创建数据 → 某人一年内的消费、工资薪水情况
# 消费按照2000-3000/月随机，工资按照5000-5500/月随机

data['c_s'] = data['consumption'] / data['salary']
print(data.head())
# 比例相对数 → 消费收入比

data['c_s'].plot.area(color = 'green',alpha = 0.5,ylim = [0.3,0.6],figsize=(8,3),grid=
# 创建面积图表达
consumption        salary
2017-01  2341.986842  5036.869326
2017-02  2789.342094  5252.001064
2017-03  2027.628915  5413.125666
2017-04  2278.996612  5227.287760
2017-05  2756.277242  5026.568710
------
         consumption       salary       c_s
```

```
2017-03  2027.628915  5413.125666  0.374576
2017-04  2278.996612  5227.287760  0.435981
2017-05  2756.277242  5026.568710  0.548342
```

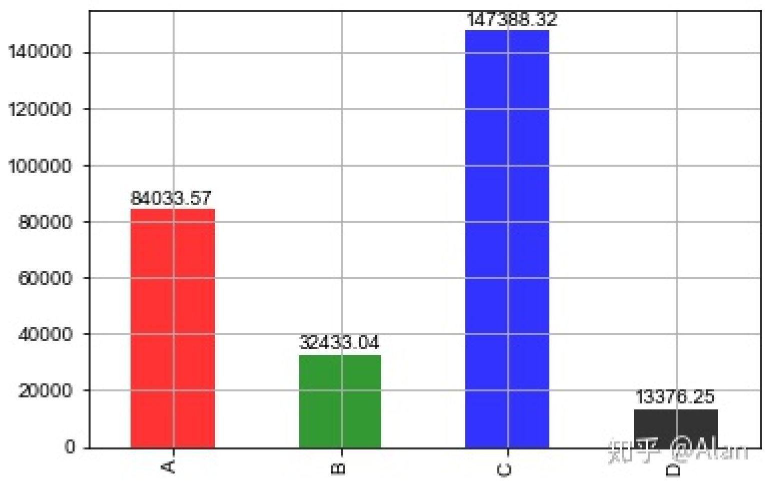
```
<matplotlib.axes._subplots.AxesSubplot at 0x11cf6fed0>
```

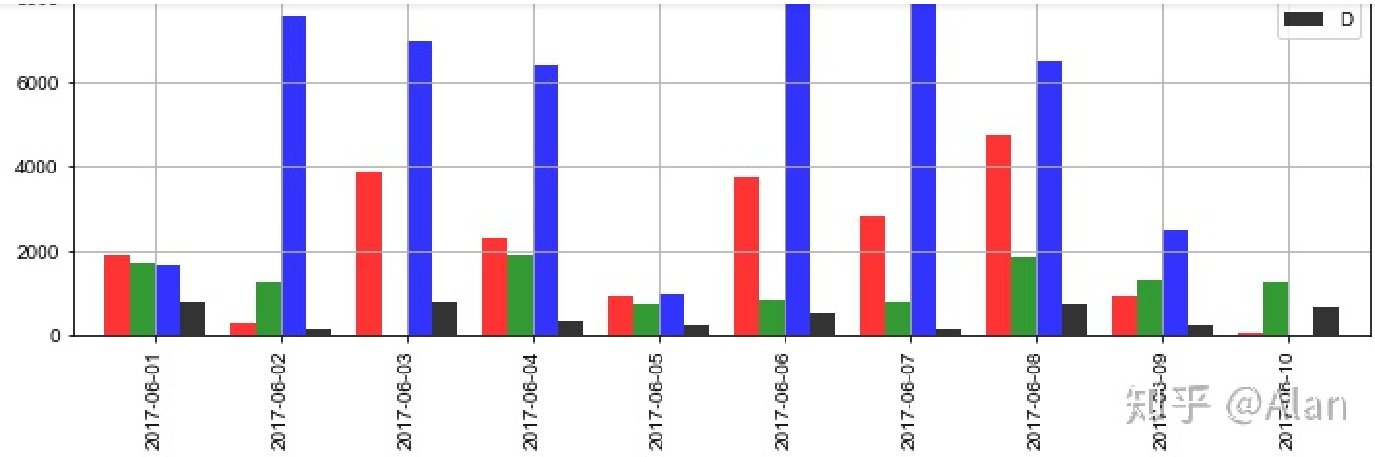



```
# 2、相对数比较 → 相除
# （3）空间比较分析（横向对比分析）
# 同类现象在同一时间不同空间的指标数值进行对比，反应同类现象在不同空间上的差异程度和现象发展不平
# 空间比较相对数 = 甲空间某一现象的数值 / 乙空间同类现象的数值
# 一个很现实的例子 → 绝对数来看，我国多经济总量世界第一，但从人均水平来看是另一回事

data = pd.DataFrame({'A':np.random.rand(30)*5000,
                     'B':np.random.rand(30)*2000,
                     'C':np.random.rand(30)*10000,
                     'D':np.random.rand(30)*800},
                    index = pd.period_range('20170601','20170630'))
print(data.head())
print('------')
# 创建数据 → 30天内A/B/C/D四个产品的销售情况
# 不同产品的销售量级不同

data.sum().plot(kind = 'bar',color = ['r','g','b','k'], alpha = 0.8, grid = True)
for i,j in zip(range(4),data.sum()):
    plt.text(i-0.25,j+2000,'%.2f' % j, color = 'k')
# 通过柱状图做横向比较 → 4个产品的销售额总量
```

```
# 关于同比与环比
# 同比 → 产品A在2015.3和2016.3的比较（相邻时间段的同一时间点）
# 环比 → 产品A在2015.3和2015.4的比较（相邻时间段的比较）
# 如何界定“相邻时间段”与“时间点”，决定了是同比还是环比
A            B            C            D
2017-06-01  1903.635299  1697.458433  1657.940621  777.840479
2017-06-02   312.175185  1247.875900  7561.101176  130.302710
2017-06-03  3856.035014    31.060706  6962.301361  779.404118
2017-06-04  2300.573349  1912.977025  6406.109791  356.947049
2017-06-05   953.273508   772.248589   988.999827  252.058645
------
```

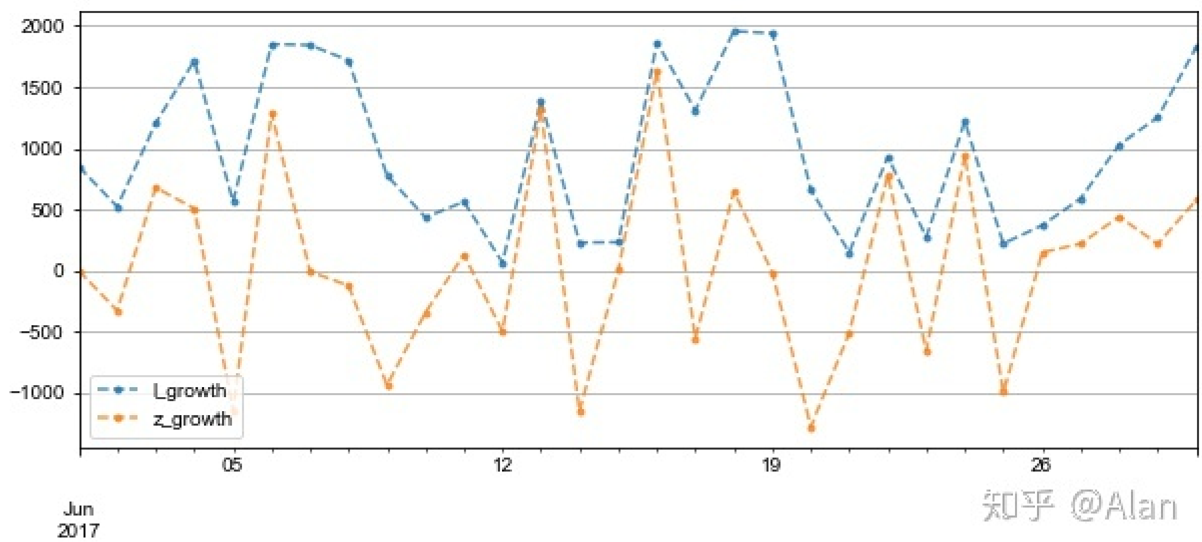
```
<matplotlib.axes._subplots.AxesSubplot at 0x11c8d4810>
```

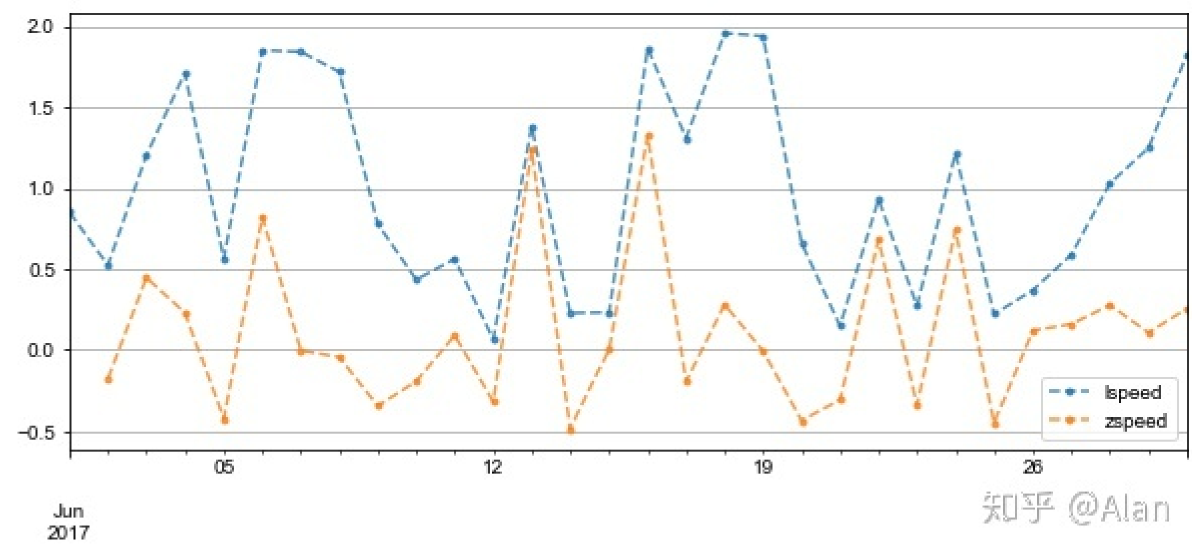





```
# 2、相对数比较 → 相除
# （4）动态对比分析（纵向对比分析）
# 同一现象在不同时间上的指标数值进行对比，反应现象的数量随着时间推移而发展变动的程度及趋势
# 最基本方法，计算动态相对数 → 发展速度
# 动态相对数（发展速度） = 某一现象的报告期数值 / 同一现象的基期数值
# 基期：用来比较的基础时期
# 报告期：所要研究的时期，又称计算期

data = pd.DataFrame({'A':np.random.rand(30)*2000+1000},
                   index = pd.period_range('20170601','20170630'))
print(data.head())
print('------')
# 创建数据 → 30天内A产品的销售情况

data['base'] = 1000  # 假设基期销售额为1000，后面每一天都为计算期
data['l_growth'] = data['A'] - data['base']  # 累计增长量 = 报告期水平 - 固定基期水平
data['z_growth'] = data['A'] - data.shift(1)['A']  # 逐期增长量 = 报告期水平 - 报告期前一
data[data.isnull()] = 0  # 替换缺失值

data[['l_growth','z_growth']].plot(figsize = (10,4),style = '--.',alpha = 0.8)

plt.legend(loc = 'lower left')
plt.grid()
# 通过折线图查看增长量情况

data['lspeed'] = data['l_growth'] / data['base']  # 定基增长速度
data['zspeed'] = data['z_growth'] / data.shift(1)['A']  # 环比增长速度
data[['lspeed','zspeed']].plot(figsize = (10,4),style = '--.',alpha = 0.8)

plt.grid()
print(data.head())
```

```
A
2017-06-01  1852.391105
2017-06-02  1523.638689
2017-06-03  2203.341280
2017-06-04  2711.615379
2017-06-05  1563.280872
------
                      A  base     l_growth       z_growth    lspeed    zspeed
2017-06-01  1852.391105  1000   852.391105       0.000000  0.852391       NaN
2017-06-02  1523.638689  1000   523.638689    -328.752415  0.523639 -0.177475
2017-06-03  2203.341280  1000  1203.341280     679.702590  1.203341  0.446105
2017-06-04  2711.615379  1000  1711.615379     508.274099  1.711615  0.230683
2017-06-05  1563.280872  1000   563.280872   -1148.334507  0.563281 -0.423487
------
```



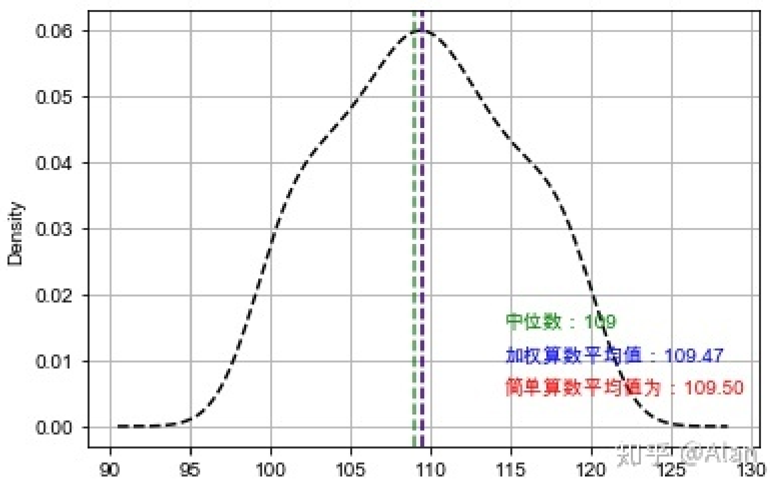


统计分析 1、统计指标对定量数据进行统计描述，常从集中趋势和离中趋势两个方面进行分析

```
2、集中趋势度量 / 离中趋势度量
# 1、集中趋势度量
# 指一组数据向某一中心靠拢的倾向，核心在于寻找数据的代表值或中心值 —— 统计平均数
# 算数平均数、位置平均数
# （1）算数平均数

data = pd.DataFrame({'value':np.random.randint(100,120,100),
                     'f':np.random.rand(100)})
data['f'] = data['f'] / data['f'].sum()  # f为权重，这里将f列设置成总和为1的权重占比
print(data.head())
print('------')
# 创建数据

mean = data['value'].mean()
print('简单算数平均值为：%.2f' % mean)
# 简单算数平均值 = 总和 / 样本数量 （不涉及权重）

mean_w = (data['value'] * data['f']).sum() / data['f'].sum()
print('加权算数平均值为：%.2f' % mean_w)
# 加权算数平均值 = (x1f1 + x2f2 + ... + xnfn) / (f1 + f2 + ... + fn)
value         f
0    110  0.017565
1    101  0.009243
2    118  0.004188
3    102  0.014161
4    109  0.008479
------
简单算数平均值为：109.50
加权算数平均值为：109.47
# 1、集中趋势度量
# （2）位置平均数

m = data['value'].mode()
print('众数为',m.tolist())
# 众数是一组数据中出现次数最多的数，这里可能返回多个值

med = data['value'].median()
print('中位数为%i' % med)
# 中位数指将总体各单位标志按照大小顺序排列后，中间位置的数字
```

```
plt.axvline(mean,color='r',linestyle="--",alpha=0.8)
plt.text(mean + 5,0.005,'简单算数平均值为：%.2f' % mean, color = 'r')
# 简单算数平均值

plt.axvline(mean_w,color='b',linestyle="--",alpha=0.8)
plt.text(mean + 5,0.01,'加权算数平均值：%.2f' % mean_w, color = 'b')
# 加权算数平均值

plt.axvline(med,color='g',linestyle="--",alpha=0.8)
plt.text(mean + 5,0.015,'中位数：%i' % med, color = 'g')
# 中位数
# **这里三个数text显示的横坐标一致，目的是图示效果不拥挤
众数为 [111]
中位数为109
```

```
Text(114.5, 0.015, '中位数：109')
```

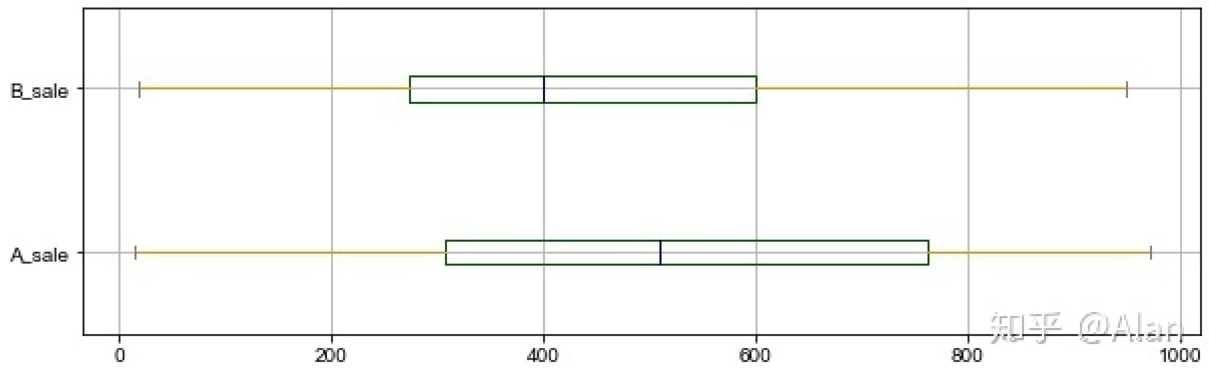



```
# 2、离中趋势度量
# 指一组数据中各数据以不同程度的距离偏离中心的趋势
# 极差与分位差、方差与标准差、离散系数

data = pd.DataFrame({'A_sale':np.random.rand(30)*1000,
                    'B_sale':np.random.rand(30)*1000},
                   index = pd.period_range('20170601','20170630'))
print(data.head())
```

```
# A/B销售额量级在同一水平
A_sale      B_sale
2017-06-01  839.793467  462.324478
2017-06-02   64.791504  530.524918
2017-06-03  721.963881   78.193191
2017-06-04  184.231508  647.775183
2017-06-05  750.953250  263.886255
------
# 2、离中趋势度量
# （1）极差、分位差

data = pd.DataFrame({'A_sale':np.random.rand(30)*1000,
                    'B_sale':np.random.rand(30)*1000},
                   index = pd.period_range('20170601','20170630'))
print(data.head())
print('------')
# 创建数据
# A/B销售额量级在同一水平

a_r = data['A_sale'].max() - data['A_sale'].min()
b_r = data['B_sale'].max() - data['B_sale'].min()
print('A销售额的极差为：%.2f, B销售额的极差为：%.2f' % (a_r,b_r))
print('------')
# 极差
# 没有考虑中间变量的变动，测定离中趋势不稳定

sta = data['A_sale'].describe()
stb = data['B_sale'].describe()
#print(sta)
a_iqr = sta.loc['75%'] - sta.loc['25%']
b_iqr = stb.loc['75%'] - stb.loc['25%']
print('A销售额的分位差为：%.2f, B销售额的分位差为：%.2f' % (a_iqr,b_iqr))
print('------')
# 分位差

color = dict(boxes='DarkGreen', whiskers='DarkOrange', medians='DarkBlue', caps='Gray'
data.plot.box(vert=False,grid = True,color = color,figsize = (10,3))
# 箱型图
A_sale      B_sale
2017-06-01  540.038536  948.523642
2017-06-02   26.021921  394.058048
2017-06-03  343.257885  372.039938
2017-06-04  714.883084  333.176287
```

```
A销售额的极差为：958.11, B销售额的极差为：930.26
------
A销售额的分位差为：455.46, B销售额的分位差为：327.09
------
```

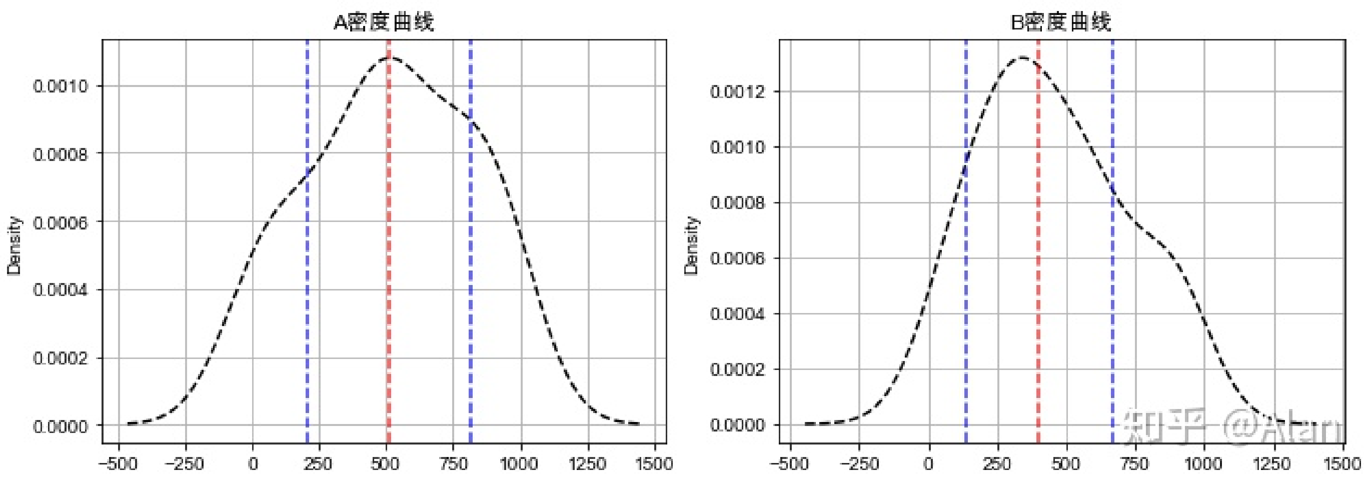
```
<matplotlib.axes._subplots.AxesSubplot at 0x11c71b990>
```




```
# 2、离中趋势度量
# （2）方差与标准差

a_std = sta.loc['std']
b_std = stb.loc['std']
a_var = data['A_sale'].var()
b_var = data['B_sale'].var()
print('A销售额的标准差为：%.2f, B销售额的标准差为：%.2f' % (a_std,b_std))
print('A销售额的方差为：%.2f, B销售额的方差为：%.2f' % (a_var,b_var))
# 方差 → 各组中数值与算数平均数离差平方的算术平均数
# 标准差 → 方差的平方根
# 标准差是最常用的离中趋势指标 → 标准差越大，离中趋势越明显

fig = plt.figure(figsize = (12,4))
ax1 = fig.add_subplot(1,2,1)
data['A_sale'].plot(kind = 'kde',style = 'k--',grid = True,title = 'A密度曲线')
plt.axvline(sta.loc['50%'],color='r',linestyle="--",alpha=0.8)
plt.axvline(sta.loc['50%'] - a_std,color='b',linestyle="--",alpha=0.8)
plt.axvline(sta.loc['50%'] + a_std,color='b',linestyle="--",alpha=0.8)
# A密度曲线，1个标准差
```

```
data['B_sale'].plot(kind = 'kde',style = 'k--',grid = True,title = 'B密度曲线')
plt.axvline(stb.loc['50%'],color='r',linestyle="--",alpha=0.8)
plt.axvline(stb.loc['50%'] - b_std,color='b',linestyle="--",alpha=0.8)
plt.axvline(stb.loc['50%'] + b_std,color='b',linestyle="--",alpha=0.8)
# B密度曲线，1个标准差
A销售额的标准差为：303.86，B销售额的标准差为：266.12
A销售额的方差为：92332.91，B销售额的方差为：70820.79
```

```
<matplotlib.lines.Line2D at 0x11d308c10>
```



4、帕累托分析

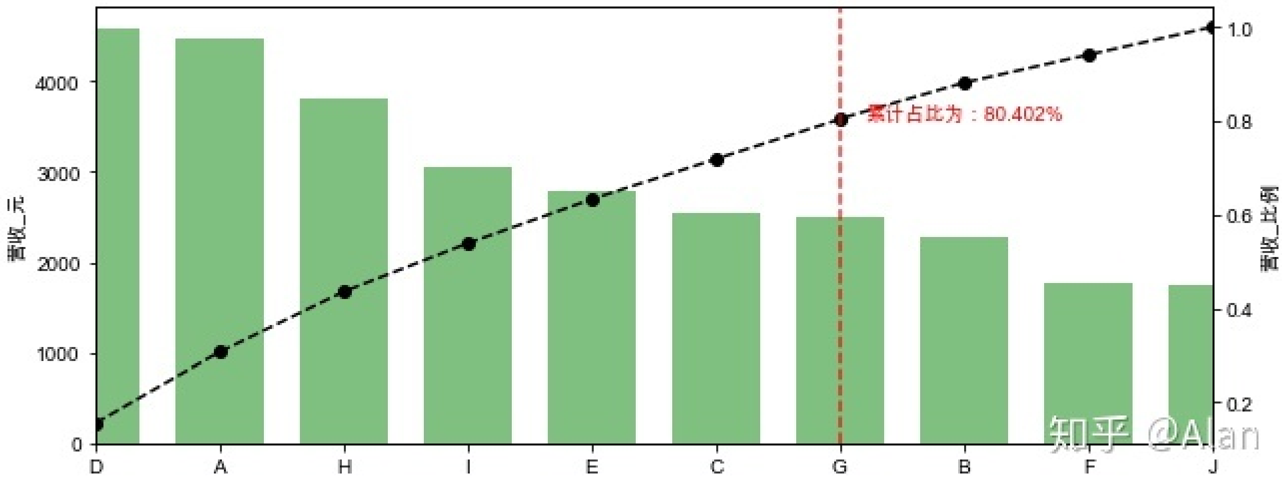
帕累托分析（贡献度分析） → 帕累托法则：20/80定律

“原因和结果、投入和产出、努力和报酬之间本来存在着无法解释的不平衡。一般来说，投入和努力可以分为两种不同的类型： 多数，它们只能造成少许的影响；少数，它们造成主要的、重大的影响。” → 一个公司，80%利润来自于20%的畅销产品，而其他80%的产品只产生了20%的利润

```
例如：

世界上大约80％的资源是由世界上15％的人口所耗尽的
世界财富的80％为25％的人所拥有；在一个国家的医疗体系中
20％的人口与20％的疾病，会消耗80％的医疗资源
```

```
# 帕累托分布分析

data = pd.Series(np.random.randn(10)*1200+3000,
                 index = list('ABCDEFGHIJ'))
print(data)
print('------')
# 创建数据，10个品类产品的销售额

data.sort_values(ascending=False, inplace= True)
# 由大到小排列

plt.figure(figsize = (10,4))
data.plot(kind = 'bar', color = 'g', alpha = 0.5, width = 0.7)
plt.ylabel('营收_元')
# 创建营收柱状图

p = data.cumsum()/data.sum()  # 创建累计占比，Series
key = p[p>0.8].index[0]
key_num = data.index.tolist().index(key)
print('超过80%累计占比的节点值索引为：' ,key)
print('超过80%累计占比的节点值索引位置为：' ,key_num)
print('------')
# 找到累计占比超过80%时候的index
# 找到key所对应的索引位置

p.plot(style = '--ko', secondary_y=True)  # secondary_y → y副坐标轴
plt.axvline(key_num,color='r',linestyle="--",alpha=0.8)
plt.text(key_num+0.2,p[key],'累计占比为：%.3f%%' % (p[key]*100), color = 'r')  # 累计占比
plt.ylabel('营收_比例')
# 绘制营收累计占比曲线

key_product = data.loc[:key]
print('核心产品为：')
print(key_product)
# 输出决定性因素产品
A    4475.042399
B    2270.872667
C    2546.500453
D    4589.334625
E    2778.845810
F    1762.224373
G    2509.467588
```

```
J     1756.489492
dtype: float64
------
超过80%累计占比的节点值索引为：  G
超过80%累计占比的节点值索引位置为：  6
------
核心产品为：
D     4589.334625
A     4475.042399
H     3803.146712
I     3049.155580
E     2778.845810
C     2546.500453
G     2509.467588
dtype: float64
```



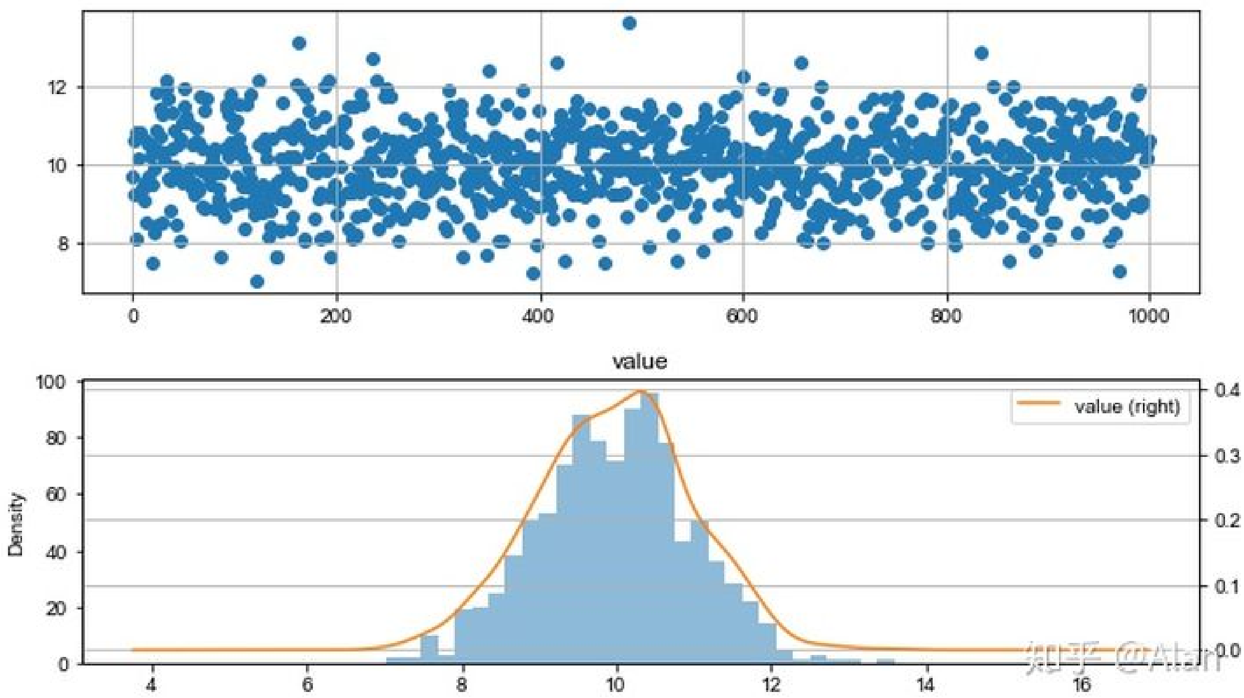
5、正态性检验

利用观测数据判断总体是否服从正态分布的检验称为正态性检验，它是统计判决中重要的一种特殊的拟合优度假设检验。

直方图初判 / QQ图判断 / K-S检验

```
# 直方图初判

s = pd.DataFrame(np.random.randn(1000)+10,columns = ['value'])
print(s.head())
```

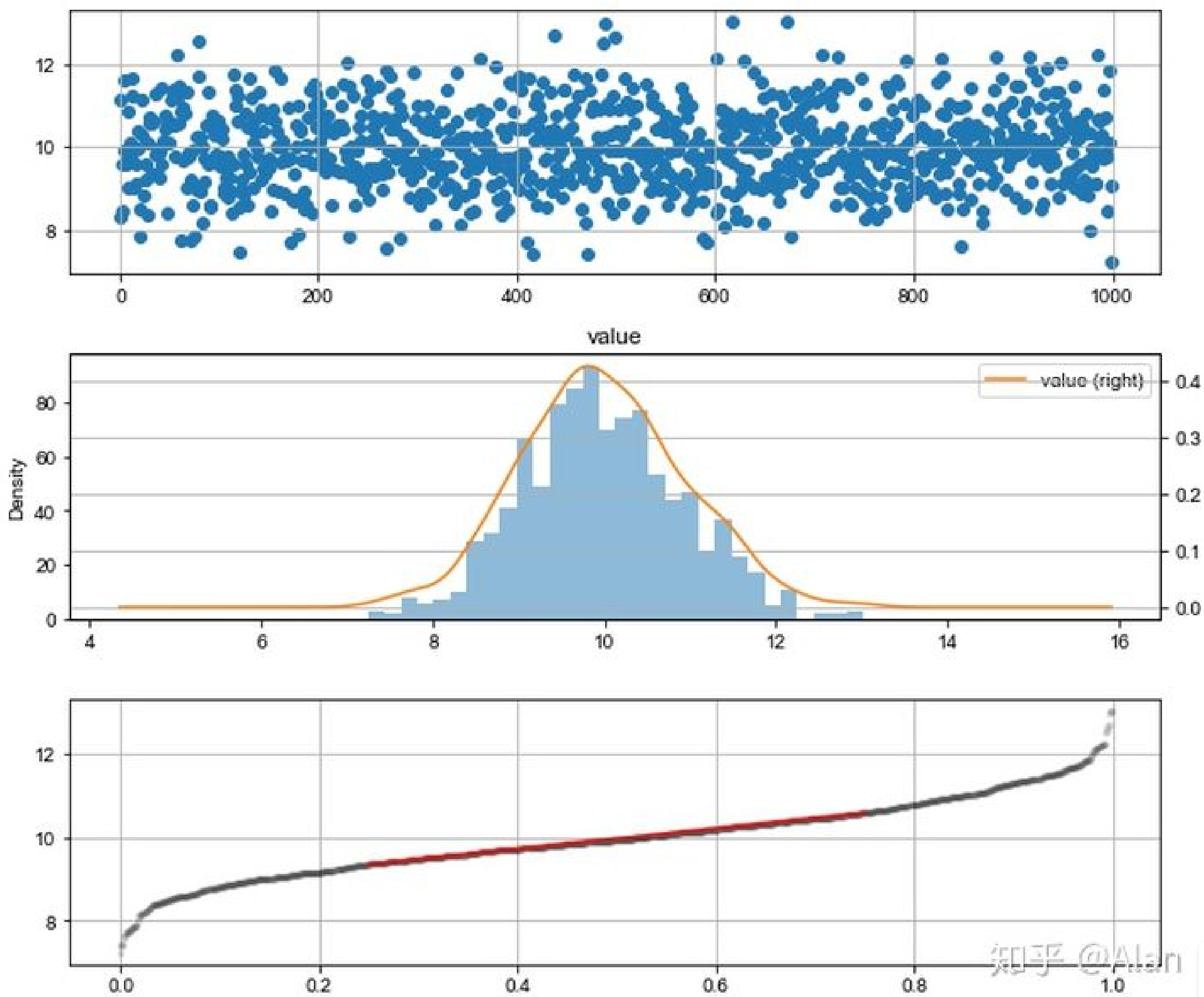
```
fig = plt.figure(figsize = (10,6))
ax1 = fig.add_subplot(2,1,1)  # 创建子图1
ax1.scatter(s.index, s.values)
plt.grid()
# 绘制数据分布图

ax2 = fig.add_subplot(2,1,2)  # 创建子图2
s.hist(bins=30,alpha = 0.5,ax = ax2)
s.plot(kind = 'kde', secondary_y=True,ax = ax2)
plt.grid()
# 绘制直方图
# 呈现较明显的正太性
value
0    9.690334
1   10.588470
2    9.228373
3   10.781879
4    8.113847
```




```
# QQ图判断
# QQ图通过把测试样本数据的分位数与已知分布相比较，从而来检验数据的分布情况

# QQ图是一种散点图，对应于正态分布的QQ图，就是由标准正态分布的分位数为横坐标，样本值为纵坐标的散
# 参考直线：四分之一分位点和四分之三分位点这两点确定，看散点是否落在这条线的附近
```

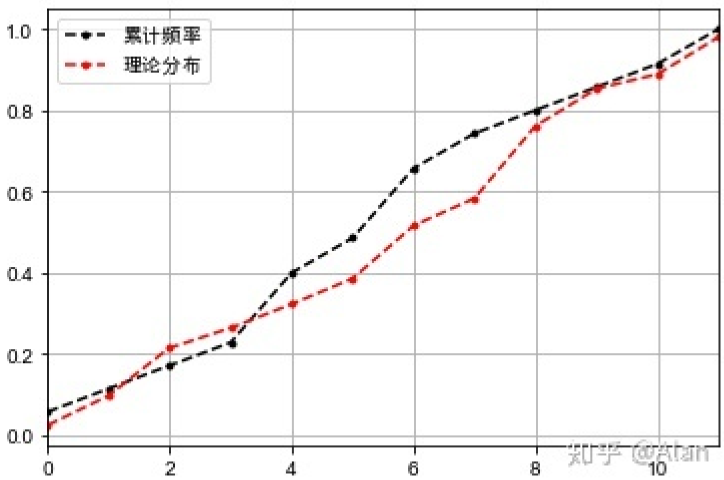
```
# ② 排序后，计算出每个数据对应的百分位p{i}，即第i个数据x(i)为p(i)分位数，其中p(i)=(i-0.5)/n
# ③ 绘制直方图 + qq图，直方图作为参考

s = pd.DataFrame(np.random.randn(1000)+10,columns = ['value'])
print(s.head())
# 创建随机数据

mean = s['value'].mean()
std = s['value'].std()
print('均值为：%.2f，标准差为：%.2f' % (mean,std))
print('------')
#  计算均值，标准差

s.sort_values(by = 'value', inplace = True)  # 重新排序
s_r = s.reset_index(drop = False)  # 重新排序后，更新index
s_r['p'] = (s_r.index - 0.5) / len(s_r)
s_r['q'] = (s_r['value'] - mean) / std
print(s_r.head())
print('------')
# 计算百分位数 p(i)
# 计算q值

st = s['value'].describe()
x1 ,y1 = 0.25, st['25%']
x2 ,y2 = 0.75, st['75%']
print('四分之一位数为：%.2f，四分之三位数为：%.2f' % (y1,y2))
print('------')
# 计算四分之一位数、四分之三位数

fig = plt.figure(figsize = (10,9))
ax1 = fig.add_subplot(3,1,1)  # 创建子图1
ax1.scatter(s.index, s.values)
plt.grid()
# 绘制数据分布图

ax2 = fig.add_subplot(3,1,2)  # 创建子图2
s.hist(bins=30,alpha = 0.5,ax = ax2)
s.plot(kind = 'kde', secondary_y=True,ax = ax2)
plt.grid()
# 绘制直方图

ax3 = fig.add_subplot(3,1,3)  # 创建子图3
ax3.plot(s_r['p'],s_r['value'],'k.',alpha = 0.1)
```

```
# 绘制QQ图，直线为四分之一位数、四分之三位数的连线，基本符合止态分布
value
0  11.163467
1   8.338891
2   9.588068
3   8.463832
4   9.915527
均值为：9.98，标准差为：0.95
------
   index     value       p         q
0    998  7.242628 -0.0005 -2.885516
1    416  7.400448  0.0005 -2.719049
2    471  7.414402  0.0015 -2.704331
3    121  7.472432  0.0025 -2.643121
4    268  7.583349  0.0035 -2.526127
------
四分之一位数为：9.35，四分之三位数为：10.58
------
```



```
# KS检验，理论推导

data = [87,77,92,68,80,78,84,77,81,80,80,77,92,86,
       76,80,81,75,77,72,81,72,84,86,80,68,77,87,
       76,77,78,92,75,80,78]
# 样本数据，35位健康男性在未进食之前的血糖浓度

df = pd.DataFrame(data, columns =['value'])
u = df['value'].mean()
std = df['value'].std()
print("样本均值为：%.2f，样本标准差为：%.2f" % (u,std))
print('------')
# 查看数据基本统计量

s = df['value'].value_counts().sort_index()
df_s = pd.DataFrame({'血糖浓度':s.index,'次数':s.values})
# 创建频率数据

df_s['累计次数'] = df_s['次数'].cumsum()
df_s['累计频率'] = df_s['累计次数'] / len(data)
df_s['标准化取值'] = (df_s['血糖浓度'] - u) / std
df_s['理论分布'] =[0.0244,0.0968,0.2148,0.2643,0.3228,0.3859,0.5160,0.5832,0.7611,0.853
df_s['D'] = np.abs(df_s['累计频率'] - df_s['理论分布'])
dmax = df_s['D'].max()
print("实际观测D值为：%.4f" % dmax)
# D值序列计算结果表格

df_s['累计频率'].plot(style = '--k.')
df_s['理论分布'].plot(style = '--r.')
plt.legend(loc = 'upper left')
plt.grid()
# 密度图表示

df_s
样本均值为：79.74，样本标准差为：5.94
------
实际观测D值为：0.1597
```

0	68	2	2	0.057143	-1.977701	0.0244	0.032743
1	72	2	4	0.114286	-1.304031	0.0968	0.017486
2	75	2	6	0.171429	-0.798779	0.2148	0.043371
3	76	2	8	0.228571	-0.630362	0.2643	0.035729
4	77	6	14	0.400000	-0.461945	0.3228	0.077200
5	78	3	17	0.485714	-0.293527	0.3859	0.099814
6	80	6	23	0.657143	0.043307	0.5160	0.141143
7	81	3	26	0.742857	0.211725	0.5832	0.159657
8	84	2	28	0.800000	0.716977	0.7611	0.038900
9	86	2	30	0.857143	1.053811	0.8531	0.004043
10	87	2	32	0.914286	1.222229	0.8888	0.025486
11	92	3	35	1.000000	2.064315	0.9803	0.019700




```
# 直接用算法做KS检验

from scipy import stats
# scipy包是一个高级的科学计算库，它和Numpy联系很密切，Scipy一般都是操控Numpy数组来进行科学计算

data = [87,77,92,68,80,78,84,77,81,80,80,77,92,86,
        76,80,81,75,77,72,81,72,84,86,80,68,77,87,
        76,77,78,92,75,80,78]
```

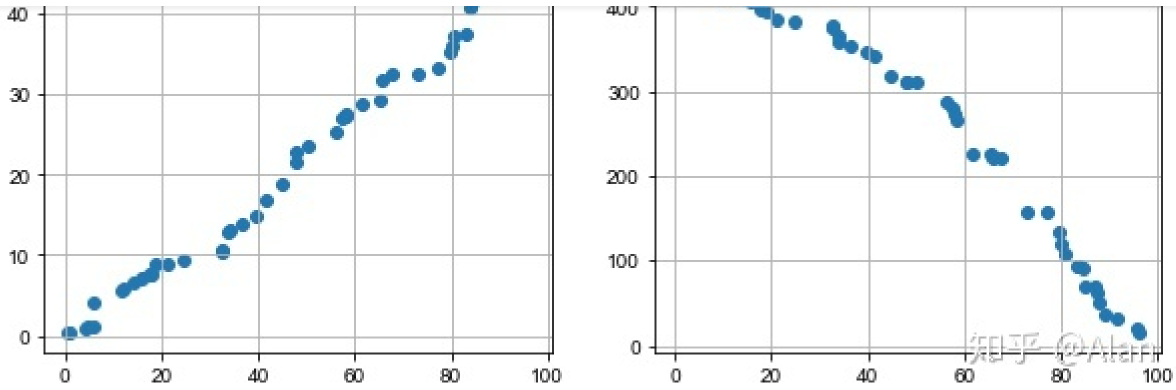
```
df = pd.DataFrame(data, columns =['value'])
u = df['value'].mean()  # 计算均值
std = df['value'].std()  # 计算标准差
stats.kstest(df['value'], 'norm', (u, std))
# .kstest方法：KS检验，参数分别是：待检验的数据，检验方法（这里设置成norm正态分布），均值与标准
# 结果返回两个值：statistic → D值，pvalue → P值
# p值大于0.05，为正态分布
KstestResult(statistic=0.1590180704824098, pvalue=0.3066297258358026)
```

6、相关性分析

1、分析连续变量之间的线性相关程度的强弱

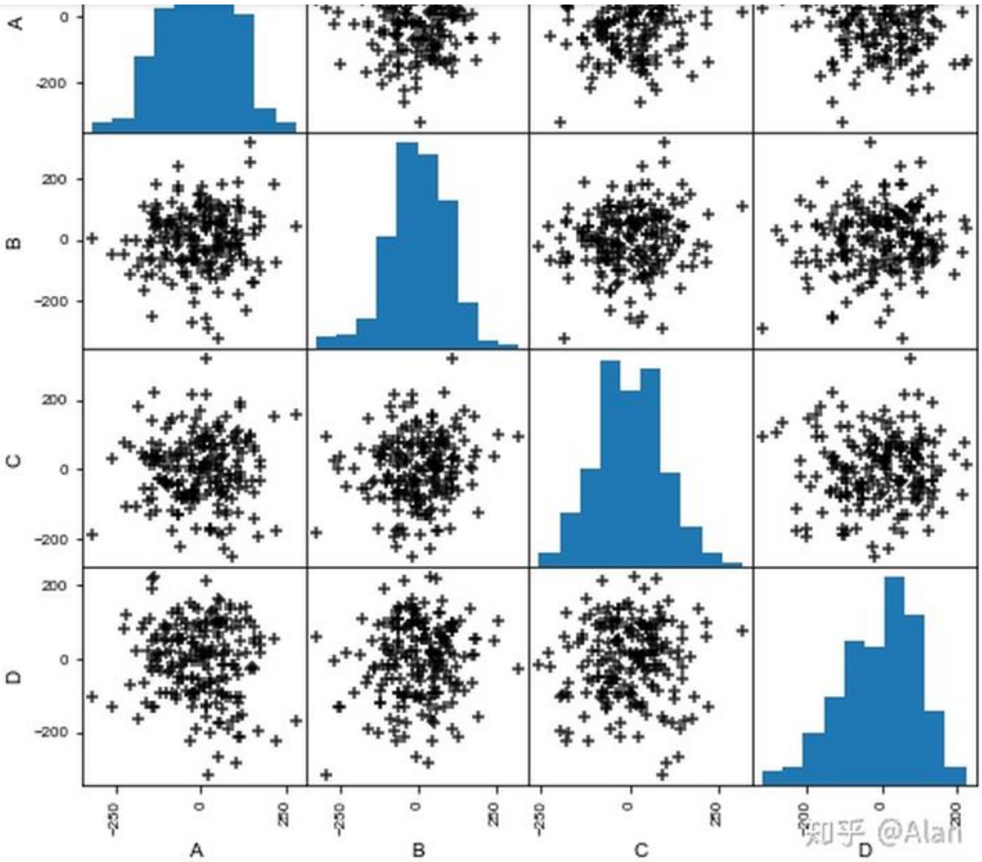
2、图示初判 / Pearson相关系数（皮尔逊相关系数） / Sperman秩相关系数（斯皮尔曼相关系数）

```
# 图示初判
# （1）变量之间的线性相关性

data1 = pd.Series(np.random.rand(50)*100).sort_values()
data2 = pd.Series(np.random.rand(50)*50).sort_values()
data3 = pd.Series(np.random.rand(50)*500).sort_values(ascending = False)
# 创建三个数据：data1为0-100的随机数并从小到大排列，data2为0-50的随机数并从小到大排列，data3为

fig = plt.figure(figsize = (10,4))
ax1 = fig.add_subplot(1,2,1)
ax1.scatter(data1, data2)
plt.grid()
# 正线性相关

ax2 = fig.add_subplot(1,2,2)
ax2.scatter(data1, data3)
plt.grid()
# 负线性相关
```




```
# 图示初判
# （2）散点图矩阵初判多变量间关系

data = pd.DataFrame(np.random.randn(200,4)*100, columns = ['A','B','C','D'])
pd.plotting.scatter_matrix(data,figsize=(8,8),#从以前的 pd.scatter_matrix 变成 pd.plott
                  c = 'k',
                 marker = '+',
                 diagonal='hist',
                 alpha = 0.8,
                 range_padding=0.1)
data.head()
```

	A	B	C	D
0	14.527118	-77.085783	214.283105	96.162098
1	27.799885	128.546147	-62.059859	-63.990536
2	23.191104	-100.435699	-22.681817	-28.584042
3	122.216444	13.379068	47.595585	-150.138677
4	177.082885	-104.635375	-31.655694	39.010097




```
# Pearson  皮尔逊相关系数

data1 = pd.Series(np.random.rand(100)*100).sort_values()
data2 = pd.Series(np.random.rand(100)*50).sort_values()
data = pd.DataFrame({'value1':data1.values,
                     'value2':data2.values})
print(data.head())
print('------')
# 创建样本数据

u1,u2 = data['value1'].mean(),data['value2'].mean()  # 计算均值
std1,std2 = data['value1'].std(),data['value2'].std()  # 计算标准差
print('value1正态性检验：\n',stats.kstest(data['value1'], 'norm', (u1, std1)))
print('value2正态性检验：\n',stats.kstest(data['value2'], 'norm', (u2, std2)))
print('------')
# 正态性检验 → pvalue >0.05

data['(x-u1)*(y-u2)'] = (data['value1'] - u1) * (data['value2'] - u2)
data['(x-u1)**2'] = (data['value1'] - u1)**2
data['(y-u2)**2'] = (data['value2'] - u2)**2
print(data.head())
print('------')
```

```
r = data['(x-u1)*(y-u2)'].sum() / (np.sqrt(data['(x-u1)**2'].sum() * data['(y-u2)**2']
print('Pearson相关系数为：%.4f' % r)
# 求出r
# |r| > 0.8 → 高度线性相关
value1    value2
0  0.581748  0.268493
1  0.812299  0.716837
2  8.655752  1.939986
3  9.425807  2.032081
4  9.534678  2.989555
------
value1正态性检验：
 KstestResult(statistic=0.09652201985215947, pvalue=0.29103031091013415)
value2正态性检验：
 KstestResult(statistic=0.06822128521434478, pvalue=0.7542249818174859)
------
     value1    value2  (x-u1)*(y-u2)    (x-u1)**2   (y-u2)**2
0  0.581748  0.268493    1307.570769  2570.533677  665.130875
1  0.812299  0.716837    1278.997004  2547.208845  642.206209
2  8.655752  1.939986    1028.091407  1817.012298  581.708744
3  9.425807  2.032081    1005.663950  1751.955912  577.274789
4  9.534678  2.989555     963.075976  1742.853827  532.181943
------
Pearson相关系数为：0.9898
# Pearson相关系数 - 算法

data1 = pd.Series(np.random.rand(100)*100).sort_values()
data2 = pd.Series(np.random.rand(100)*50).sort_values()
data = pd.DataFrame({'value1':data1.values,
                     'value2':data2.values})
print(data.head())
print('------')
# 创建样本数据

data.corr()
# pandas相关性方法：data.corr(method='pearson', min_periods=1) → 直接给出数据字段的相关系
# method默认pearson
```

```
0   0.349378  0.108425
1   2.347325  0.276524
2   3.604632  0.357123
3   4.365481  0.521398
4   4.690893  0.757848
------
```

	value1	value2
value1	1.00000	0.99131
value2	0.99131	1.00000


```
# Sperman秩相关系数

data = pd.DataFrame({'智商':[106,86,100,101,99,103,97,113,112,110],
                    '每周看电视小时数':[7,0,27,50,28,29,20,12,6,17]})
print(data)
print('------')
# 创建样本数据

data.sort_values('智商', inplace=True)
data['range1'] = np.arange(1,len(data)+1)
data.sort_values('每周看电视小时数', inplace=True)
data['range2'] = np.arange(1,len(data)+1)
print(data)
print('------')
# “智商”、“每周看电视小时数”重新按照从小到大排序，并设定秩次index

data['d'] = data['range1'] - data['range2']
data['d2'] = data['d']**2
print(data)
print('------')
# 求出di，di2

n = len(data)
rs = 1 - 6 * (data['d2'].sum()) / (n * (n**2 - 1))
```

```
   智商  每周看电视小时数
0  106         7
1   86         0
2  100        27
3  101        50
4   99        28
5  103        29
6   97        20
7  113        12
8  112         6
9  110        17
------
    智商  每周看电视小时数  range1  range2
1   86         0       1       1
8  112         6       9       2
0  106         7       7       3
7  113        12      10       4
9  110        17       8       5
6   97        20       2       6
2  100        27       4       7
4   99        28       3       8
5  103        29       6       9
3  101        50       5      10
------
    智商  每周看电视小时数  range1  range2  d  d2
1   86         0       1       1  0   0
8  112         6       9       2  7  49
0  106         7       7       3  4  16
7  113        12      10       4  6  36
9  110        17       8       5  3   9
6   97        20       2       6 -4  16
2  100        27       4       7 -3   9
4   99        28       3       8 -5  25
5  103        29       6       9 -3   9
3  101        50       5      10 -5  25
------
Pearson相关系数为: -0.1758
# Pearson相关系数 - 算法

data = pd.DataFrame({'智商':[106,86,100,101,99,103,97,113,112,110],
                    '每周看电视小时数':[7,0,27,50,28,29,20,12,6,17]})
print(data)
print('------')
```

```
data.corr(method='spearman')
# pandas相关性方法：data.corr(method='pearson', min_periods=1) → 直接给出数据字段的相关系
# method默认pearson
智商  每周看电视小时数
0  106          7
1   86          0
2  100         27
3  101         50
4   99         28
5  103         29
6   97         20
7  113         12
8  112          6
9  110         17
------
```

	智商	每周看电视小时数
智商	1.000000	-0.175758
每周看电视小时数	-0.175758	1.000000

发布于 2020-01-29

特征工程　数据分析　统计学

文章被以下专栏收录

推荐阅读