Python 简单数据相关性分析

我们在上一期 《Python入坑实践 tushare入门》 介绍了如何使用 tushare 获取金融数据。那么，在获取数据之后，我们会有对数据进行分析的需要。这一期我们将来尝试一下使用 Python 进行简单的数据相关性分析。我们将先了解相关性分析的一些数学基础，然后我们会使用 Python 自己去实现这些数学的计算方法，最后我们会使用 numpy 和 pandas 来验证我们自己手写的计算函数是否正确。

目标:

- 了解相关性分析的基本数学概念
- 自己编写计算函数
- 掌握 numpy 以及 pandas 的相关系数运算方法

我们以下的演示，使用的环境是 树莓派3B Raspbian 9.4，Berryconda 4.5.11 环境，Python 3.6.1，Python 库依赖： pandas、numpy。

在开始内容之前，我们先来创建实验数据，实验数据使用列表存储，数据为 [0, 100] 区间随机生成的 20 个数据。

In [1]:

```
import random
a = [random.randint(0, 100) for a in range(20)]
b = [random.randint(0, 100) for b in range(20)]
print(a)
print(b)
```

```
[77, 1, 85, 90, 65, 40, 98, 62, 40, 69, 20, 74, 75, 34, 86, 25, 46, 34, 59, 5]
[87, 4, 33, 4, 16, 62, 4, 65, 78, 36, 60, 53, 87, 81, 0, 28, 93, 94, 99, 38]
```

数学基础部分

在开始相关性分析之前，我们需要先了解两个数学统计的基本概念:

- 期望
- 离散度

期望
期望这一词可能大家不是非常熟悉，但是如果说平均数，或者均值，应该就非常了解了。期望是描述一组数据的中心倾向的一个指标之一，在计算期望的时候，对于连续变量和离散变量，我们的计算方法是不一样的，以下分别是连续变量和离散变量的期望计算公式:

对于连续随机变量 $E[X] = \int_{\pi} XdP$

对于离散随机变量 $E[X] = \sum_i p_i x_i$

在一般情况下，我们通过实验或者调查统计获取的数据很大一部分都属于离散随机变量，那么这里的期望我们也可以简单的理解为平均数，那么既然是平均数，那么我们就可以非常简单编写一个计算离散变量的期望的函数了。

In [2]:

```
def mean(x):
  return sum(x) / len(x)
mean_a = mean(a)
mean_b = mean(b)
print('a组数据期望： ', mean_a)
print('b组数据期望： ', mean_b)
```

```
a组数据期望：  54.25
b组数据期望：  51.1
```

方差&标准差

方差是用来描述一组数据的离散程度的指标，标准差即方差的开方。对于方差，和均值期望一样，对于连续变量和离散变量也是有不同的计算公式的，以下分别是对于连续变量以及离散变量的方差计算公式：

对于连续随机变量 $Var(X) = \int (x-\mu)^2 f(x) dx = \int x^2 f(x) dx - \mu^2$

对于离散随机变量 $Var(X) = E[(X-\mu^2)] = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2$

我们在使用 Python 做一些简单的数据分析时，一般离散变量的情况比较多，所以我们现在就只考虑离散变量，以下我们开始编写我们的方差以及标准差的 Python 函数：

In [3]:

```
# 计算每一项数据与均值的差
def de_mean(x):
  x_bar = mean(x)
  return [x_i - x_bar for x_i in x]
# 辅助计算函数 dot product 、sum_of_squares
def dot(v, w):
  return sum(v_i * w_i for v_i, w_i in zip(v, w))
def sum_of_squares(v):
  return dot(v, v)
# 方差
def variance(x):
  n = len(x)
  deviations = de_mean(x)
  return sum_of_squares(deviations) / (n - 1)
# 标准差
import math
def standard_deviation(x):
  return math.sqrt(variance(x))

var_a = variance(a)
var_b = variance(b)
print('a组数据的方差： ', var_a)
print('b组数据的方差： ', var_b)
std_a = standard_deviation(a)
std_b = standard_deviation(b)
print('a组数据的标准差： ', std_a)
print('b组数据的标准差： ', std_b)
```

```
a组数据的方差：  814.9342105263158
b组数据的方差：  1170.5157894736844
a组数据的标准差：  28.547052571610887
b组数据的标准差：  34.21280154377429
```

协方差、相关系数计算

接下来，我们开始进入相关性分析的计算。我们一般可以使用相关系数来衡量两组数据的相关性，相关系数的取值会在 [-1, 1]，其中 -1 代表完全负相关，1代表完全相关。同样的，下面给出协方差以及相关系数的计算公式：

协方差 $cov(X,Y) = E[(X-\mu)(Y-\upsilon)] = E(X \cdot Y) - \mu\upsilon$

相关系数 $\eta = \frac{cov(X,Y)}{\sqrt{var(X) \cdot var(Y)}}$

同样的，我们根据上面的公式，使用 Python 编写我们的计算函数：

In [4]:

```
# 协方差
def covariance(x, y):
  n = len(x)
  return dot(de_mean(x), de_mean(y)) / (n -1)
# 相关系数
def correlation(x, y):
  stdev_x = standard_deviation(x)
  stdev_y = standard_deviation(y)
  if stdev_x > 0 and stdev_y > 0:
    return covariance(x, y) / stdev_x / stdev_y
  else:
    return 0

cov_a_b = covariance(a, b)
corr_a_b = correlation(a, b)
print('ab的协方差： ', cov_a_b)
print('ab的相关系数： ', corr_a_b)
```

```
ab的协方差：  -180.18421052631575
ab的相关系数：  -0.18448744582377608
```

numpy 以及 pandas 相关性分析的运用

在实际的使用中，我们往往不需要自己编写以上的函数，我们可以使用 numpy 或者 pandas 库来完成相同的工作，使用 numpy 或者 pandas 还有另外的好处，就是这些库运行计算的速度会比我们自己编写的函数快很多，尤其是在处理大规模数据的时候更为明显，因此，无论是为了方便还是效率，我们都应该直接使用现成的库而不是自己重复造轮子。上面自己编写的函数只是为了了解相关性分析的概念。

numpy

In [5]:

```
import numpy as np
# 先构造一个矩阵
ab = np.array([a, b])
# 计算协方差矩阵
np_cov_a_b = np.cov(ab)
# 计算相关系数
np_corr_a_b = np.corrcoef(ab)
print('numpy计算ab协方差矩阵：')
print(np_cov_a_b)
print('numpy计算ab相关矩阵：')
print(np_corr_a_b)
```

```
numpy计算ab协方差矩阵：
[[ 814.93421053 -180.18421053]
 [-180.18421053 1170.51578947]]
numpy计算ab相关矩阵：
[[ 1.          -0.18448745]
 [-0.18448745  1.        ]]
```

pandas

In [6]:

```
import pandas as pd
# 使用 DataFrame 作为数据结构，为方便计算，我们会将 ab 矩阵转置
dfab = pd.DataFrame(ab.T, columns=['A', 'B'])
# 计算 a b 协方差
pd_cov_a_b = dfab.A.cov(dfab.B)
# 计算 a b 相关系数
pd_corr_a_b = dfab.A.corr(dfab.B)
print('pandas计算ab协方差：')
print(pd_cov_a_b)
print('pandas计算ab相关系数：')
print(pd_corr_a_b)
```

```
pandas计算ab协方差：
-180.18421052631575
pandas计算ab相关系数：
-0.1844874458237761
```