统计分析方法第一章作业

16337062 冯镇轩

写在前面

这次作业主要使用了pandas库进行统计，同时还使用了numpy库和scipy库的stats模块。画图使用的是matplotlib和pylab库。下面是程序头部分。

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import pylab
import os
import time
from multiprocessing import Process, Pool
```

一、求股票000001（股票代码）的历史股价的日均值（所有天数的股价求平均）、中位数、0.25分位数、0.75分位数，方差，标准差，变异系数，极差，四分位极差，偏度，峰度。

```
#part1
data_000001 = pd.read_csv('./data_selected/000001.csv')
f = open("./answer1.txt", 'w')
#中位数 25% 50% 标准差 日均值 75%
price_000001 = (data_000001['high'] + data_000001['low']) / 2;
result = price_000001.describe(include = [np.number])
#方差
result.loc['var'] = price_000001.var()
#极差 变异系数 四分位极差 偏度 峰度
result.loc['range'] =  result.loc['max'] - result.loc['min']
result.loc['cv'] = result.loc['std'] / result.loc['mean']
result.loc['quartile_deviation'] = result.loc['75%'] - result.loc['25%']
result.loc['skew'] = result.skew()
result.loc['kurt'] = result.kurt()
print(result, file = f)
f.close()
```

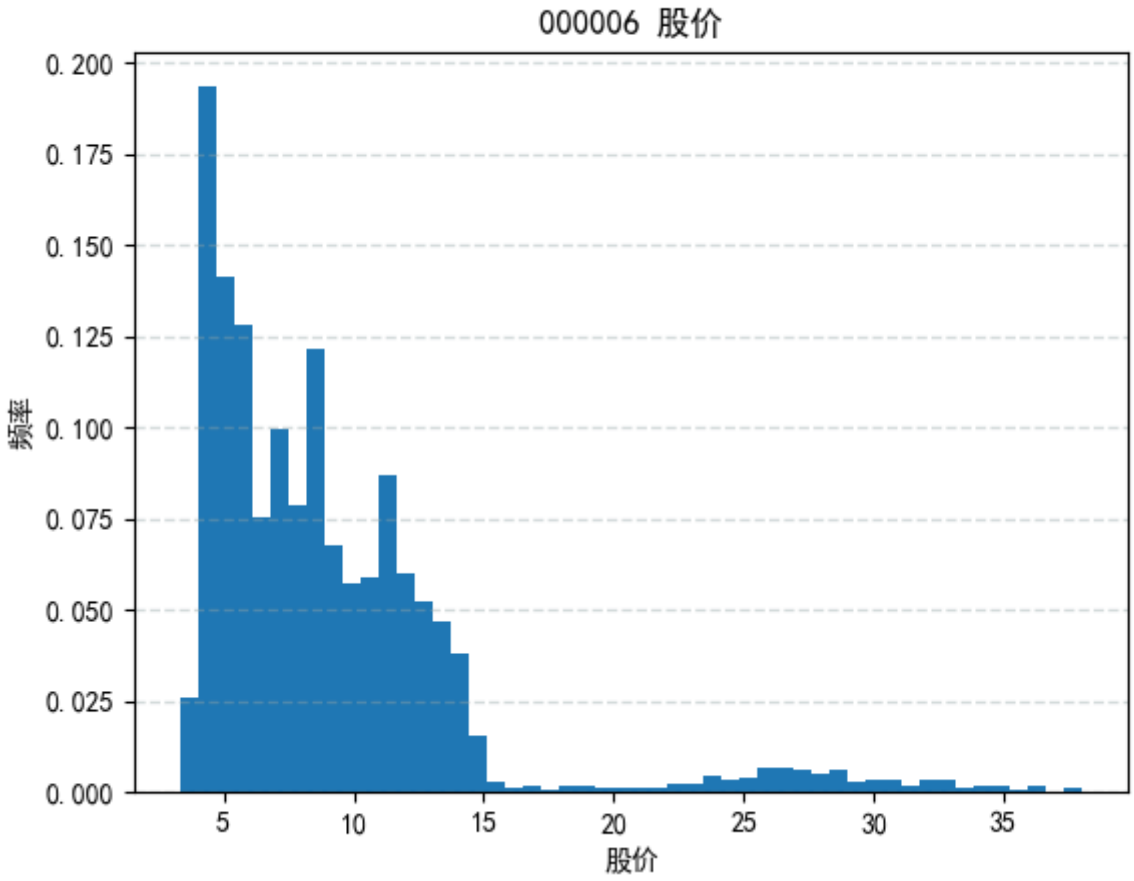
输出结果保存在"answer1.txt"中，如下所示

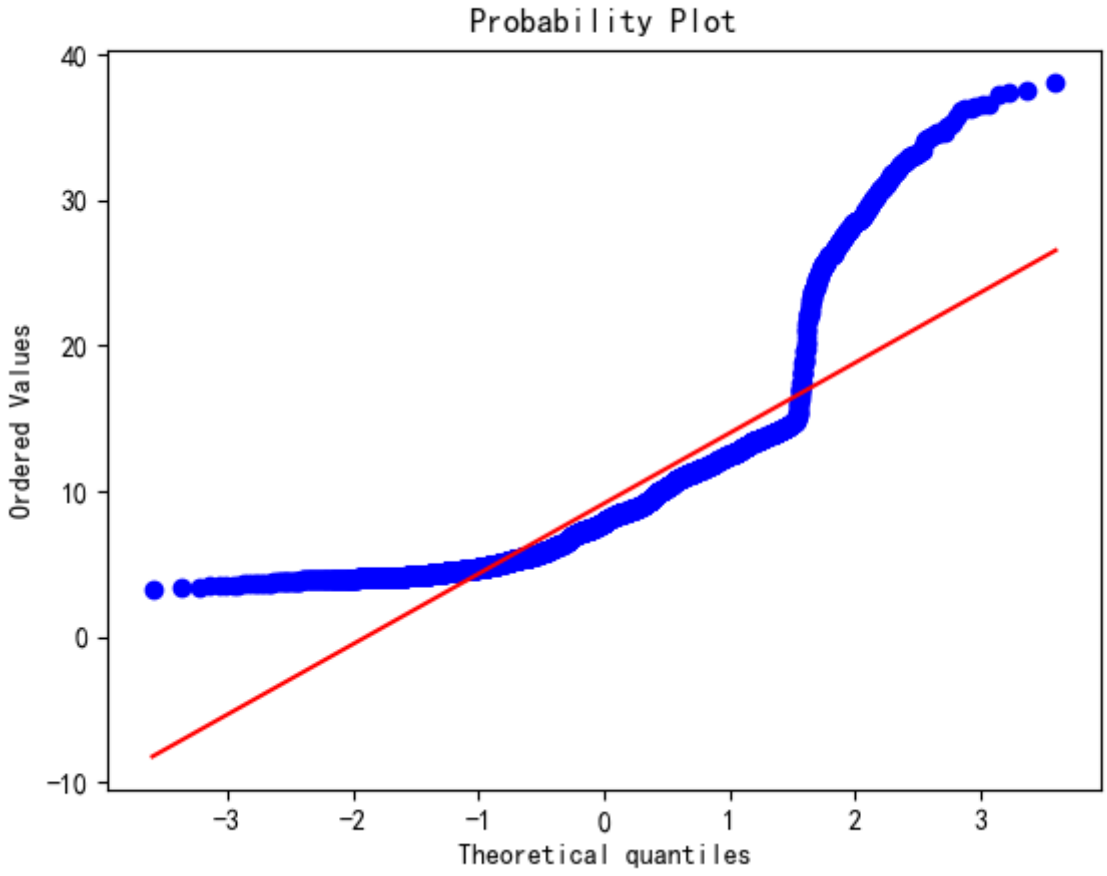
```
mean                  14.27098
std                    6.59423
min                    5.14500
25%                    9.98500
50%                   12.82501
75%                   16.78500
max                   47.99500
var                   43.48392
range                 42.85000
cv                     0.46207
quartile_deviation     6.80000
skew                   3.46303
kurt                  12.99395
```

二、对股票000006股价进行分析，选取合适组距，进行统计，画出的直方图（价格-频率）和正态QQ图，直观判断数据是否来自正态分布总体，给出简要的判断依据。如果对000006股价的差值（相邻两个日期的股价差值，忽略缺失日期，例如有t_1, t_3, t_4,则差值为: t_3- t1, t_4 - t_3），同理计算差值的直方图和正态QQ图，判断差值是否服从正态分布，给出简要的判断依据。
首先画出000006股价的频率直方图和QQ图

```
#part2
data_000006 = pd.read_csv('./data_selected/000006.csv')
#part2.1
price_000006 = (data_000006['high'] + data_000006['low'])/2;
plt.hist(price_000006, bins = 50, density = True)
plt.title('000006 股价')
plt.xlabel('股价')
plt.ylabel('频率')
plt.savefig('2.1.png')
plt.close()
#part2.2
stats.probplot(price_000006, dist="norm", plot=pylab)
pylab.savefig('2.2.png')
pylab.close()
```

得到下面两图



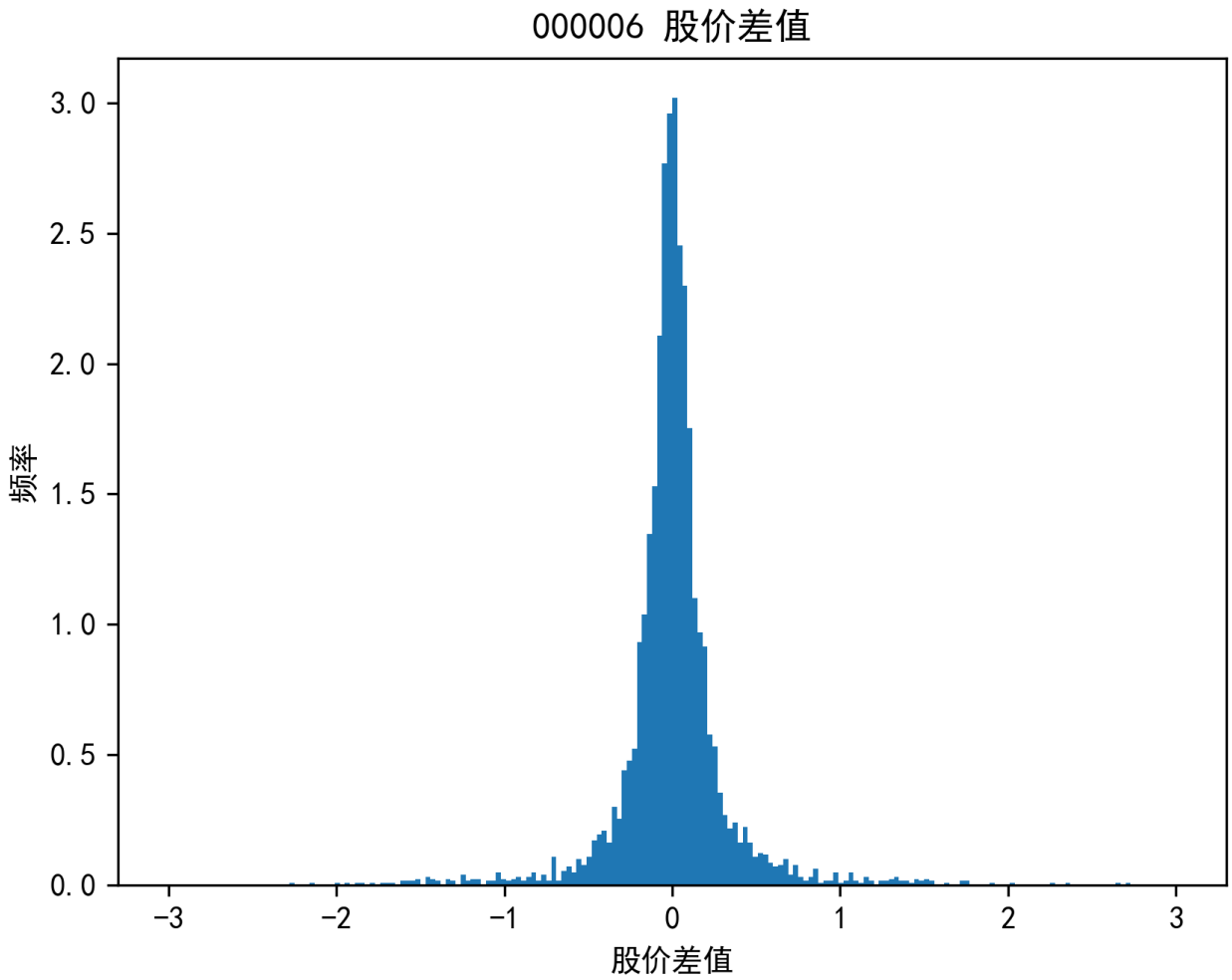


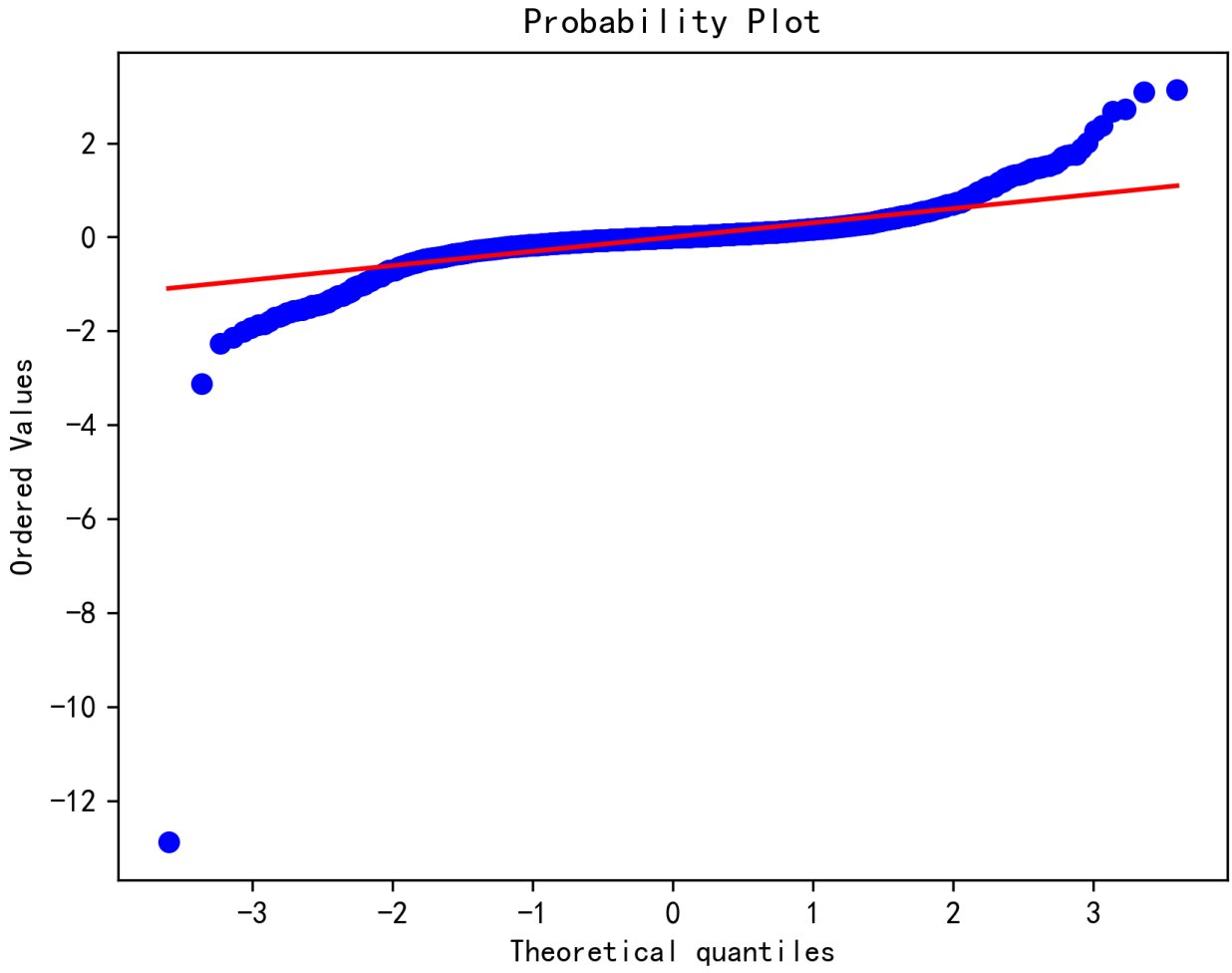
由直方图和QQ图，我认为i股票000006的股价不是来自正态分布总体。首先股价的直方图直观上不符

合正态分布的形状，其次其正态QQ图并不能很好地拟合直线。
下面画出000006股价的差值的频率直方图和QQ图

```
#part2.3
plt.rcParams['savefig.dpi'] = 300 #图片像素
plt.rcParams['figure.dpi'] = 300 #分辨率
plt.hist(price_000006.diff().dropna(), range=(-3,3), bins=200, density=True)
plt.title('000006 股价差值')
plt.xlabel('股价差值')
plt.ylabel('频率')
plt.savefig('2.3.png')
plt.close()
#part2.4
stats.probplot(price_000006.diff().dropna(), dist="norm", plot=pylab)
pylab.savefig('2.4.png')
```

得到下面两图





可以看到000006股价的差值的分布符合正态分布，QQ图与直线拟合效果也相当好，所以可以认为000006股价的差值的分布服从正态分布

三、对股票000012进行分析，求股价和成交量的Pearson，Spearman相关系数。

```
#part3
data_000012 = pd.read_csv('./data_selected/000012.csv')
data_000012['price'] = (data_000012['high'] + data_000012['low'])/2
data_000012 = data_000012[['price', 'volume']]

f = open("./answer3.txt", 'w')
print("000012的股价和成交量的Pearson相关系数\n", data_000012.corr(), "\n\n", "000012的股价和成交量
Spearman相关系数\n", data_000012.corr('spearman'), file = f)
f.close()
```

输出结果保存在"answer3.txt"中，如下所示

```
000012的股价和成交量的Pearson相关系数
          price  volume
price  1.00000 0.02982
volume 0.02982 1.00000

 000012的股价和成交量的Spearman相关系数
           price   volume
price   1.00000 -0.01870
volume -0.01870  1.00000
```

四、按照日期，对股票000001和股票000006的股价进行相关分析。例如股票000001在t_1, t_2, t_4, t_5四个日期有记录x_1, x_2, x_4, x_5; 股票000006在 t_2, t_3, t_4三个日期有记录y_2, y_3, y_4,那么我们选取有共同日期记录的值,t_2,t_4两个日期的记录，即(x_2, y2)和(x_4,y_4)进行相关分析，而丢掉缺失数据（即t_1, t_3, t_5日期的数据）。推广之，对100支股票两两进行分析，求100支不同股票股价的Pearson，Spearman相关矩阵（100×100）。 根据相关矩阵，给出这100只股票中，相关性最强(绝对值接近1) 的5对股票和相关性弱（绝对值最接近0）的5对股票，根据10支股票，求相关性假设的p值。(注意，Pearson，Spearman矩阵的元素排列依照股票代码，即，000001,000006000012，...，000717)。

我首先定义了函数cal_matrix，函数的功能是计算Pearson和Spearman相关矩阵的一部分，计算的范围是[lower_bound, upper_bound]，如[50:75]，就是计算第50支到第75支股票与其他股票的相关系数。具体操作是首先将两只股票的csv文件读入，将日期作为主键合并两个csv，再分别计算两支股票的股价并计算其相关系数，并存入数列供后面排序使用，计算完毕后将结果保存为csv，函数返回Pearson和Spearman相关系数的数列。

```
def cal_matrix(Data, lower_bound, upper_bound, file_index, Process_index):
    start_time = time.time()
    result_pearson = np.zeros((upper_bound - lower_bound, upper_bound))
    result_pearson_array = []
    result_spearman = np.zeros((upper_bound - lower_bound, upper_bound))
    result_spearman_array = []

    for x in range(0, upper_bound - lower_bound):
        for y in range(0, x + lower_bound):
            union = pd.merge(Data[x + lower_bound], Data[y], on="date", suffixes=('_left', '_ri
            union['left_price'] = (union['high_left'] + union['low_left'])/2
            union['right_price'] = (union['high_right'] + union['low_right']) /2
            union = union[['left_price', 'right_price']]
            result_pearson[x][y] = union.corr()['left_price']['right_price']
            result_pearson_array.append(abs(result_pearson[x][y]))
            result_spearman[x][y] = union.corr('spearman')['left_price']['right_price']
            result_spearman_array.append(abs(result_spearman[x][y]))

    temp = pd.DataFrame(result_pearson, index = file_index[lower_bound:upper_bound],
    columns= file_index[:upper_bound])
    temp.to_csv('Pearson' + str(Process_index) + '.csv')
    temp = pd.DataFrame(result_spearman, index = file_index[lower_bound:upper_bound],
    columns=file_index[:upper_bound])
    temp.to_csv('Spearman' + str(Process_index) + '.csv')
    end_time = time.time()
    print("Process " , Process_index, " run ", end_time- start_time,  " s\n" )
    return [result_pearson_array, result_spearman_array]
```

在主函数部分，由于 Python 的 Global interpreter lock存在，导致 Python 不能进行并行编程，只能进行并发编程。在这里我将并发进程数设为4，即将相关矩阵分成四部分进行计算。

```
    record = []
    lower_bound = [0, 50, 70, 87]
    upper_bound = [50, 70, 87, 100]
    #并发计算
    pool = Pool(processes=4)
    for i in range(4):
        record.append(pool.apply_async(cal_matrix, args=(L, lower_bound[i], upper_bound[i], fi
    pool.close()
    pool.join()
```

之后将输出的csv进行合并，结果保存在"Pearson.csv"和"Spearman.csv"中。
然后对Pearson和Spearman相关系数进行计算，取最高和最低的五个系数并计算对应股票的股价的p值。输出结果保存在"answer4.txt"中，如下所示

由Pearson系数矩阵可得，相关性最强的5对股票的Pearson相关系数及其p值为：

股票A	股票B	Pearson系数	p值
000069	000046	0.9494846389	0.00000
000046	000006	0.9464481050	0.00000
000567	000025	0.9270517821	0.00000
000069	000006	0.9150686815	0.00000
000708	000059	0.9061597117	0.00000

由Pearson系数矩阵可得，相关性最弱的5对股票的Pearson相关系数及其p值为：

股票A	股票B	Pearson系数	p值
000632	000525	-0.0002854860	0.98493
000090	000049	0.0006476817	0.96622
000661	000521	-0.0008275803	0.95648
000661	000601	0.0010438976	0.94526
000425	000036	-0.0016803145	0.91267

由Spearman系数矩阵可得，相关性最强的5对股票的Spearman相关系数及其p值为：

股票A	股票B	Spearman系数	p值
000661	000028	0.9615927289	0.00000
000567	000025	0.9431301024	0.00000
000418	000025	0.9360251005	0.00000
000661	000423	0.9311030785	0.00000
000702	000421	0.9308496460	0.00000

由Spearman系数矩阵可得，相关性最强的5对股票的Spearman相关系数及其p值为：

股票A	股票B	Spearman系数	p值
000667	000567	-0.0003678320	0.98060
000538	000078	-0.0003988047	0.97923
000088	000001	-0.0006760492	0.96482
000667	000418	0.0007418725	0.96119
000544	000425	-0.0013241253	0.93133

总结

这次作业主要使用了python中的pandas库，pandas库是一个非常高效的数据科学库。另外在第四题计算相关矩阵的时候我使用了python的并发编程，经过对比，四个核心的并发编程大概减少了一半的程序运行时间，主要是数据的合并需要比较多的操作。同时，并发编程使得代码量增加了大概一倍，产生了代码冗余，debug也变得更加困难，说明并发编程也是一把双刃剑。