

Python 金融数据分析 实验报告

(2022 春季学期-期末)

班级： 工商管理 201 班

学号： 20451040205

姓名： 何贝尔

上海商学院

2022 年 4 月 28 日

一. 实验目的及要求

实验目的：

会通过网络数据接口获取金融数据,并对获取的数据进行预处理。通过实验,掌握使用 Python 对金融数据初步分析的基本方法。

实验要求：

请根据本学期课程所学内容,完成下列任务。

(1) 通过 Tushare 数据接口 (Pro 版) 获取股票列表中**上市公司的列表** (stock_basic 接口), 列表中至少需要包含以下信息: 股票名称 (name), 地区 (area), 行业 (industry), 交易所 (exchange), 股票代码 (ts_code)

(2) 从上市公司列表中筛选出**同学自己家乡省份/市**的股票列表 (提示: 用 groupby), 并从中任意选取五支股票

(3) 获取 2022 年 04 月 22 日所有上市股票的基本面数据, 需包括动态市盈率 (pe)、每股收益 (eps)、市净率 (pb)、地域 (area)、股票代码 (ts_code)、股票名称 (name)、行业 (industry) (提示: bak_basic 接口, 参考 tushare 网站基础数据中的备用列表)

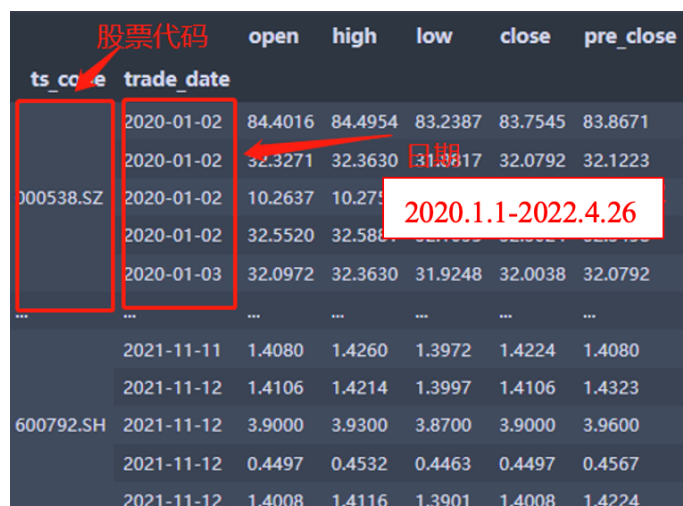
(4) 删除 DataFrame 中 industry 列

(5) 将股票代码设置为 index, 从获取的 DataFrame 中选取并查看所选取五支股票的 pe、eps、pb 数据 (提示: 用 loc)

(6) 删除上述五只股票中 eps 值最小的一支股票数据 (剩四支), 形成新的 DataFrame

(7) 从任意免费金融数据接口 (建议 Tushare Pro 版) 获取所选出的四支股票 2020 年 1 月 1 日到 2022 年 04 月 22 日的**前复权**的行情数据; (提示: 通用行情接口, pro_bar)

(8) 清洗、整理所获取的数据, 如下图形式 (包括将股票代码和交易时间设置为双重索引 Multiindex, 将 index 转变为 DatetimeIndex, 时间排序等)



股票代码	trade_date	open	high	low	close	pre_close
000538.SZ	2020-01-02	84.4016	84.4954	83.2387	83.7545	83.8671
000538.SZ	2020-01-02	32.3271	32.3630	31.3817	32.0792	32.1223
000538.SZ	2020-01-02	10.2637	10.275	10.2637	10.2637	10.2637
000538.SZ	2020-01-02	32.5520	32.588	32.5520	32.5520	32.5520
000538.SZ	2020-01-03	32.0972	32.3630	31.9248	32.0038	32.0792
...
600792.SH	2021-11-11	1.4080	1.4260	1.3972	1.4224	1.4080
600792.SH	2021-11-12	1.4106	1.4214	1.3997	1.4106	1.4323
600792.SH	2021-11-12	3.9000	3.9300	3.8700	3.9000	3.9600
600792.SH	2021-11-12	0.4497	0.4532	0.4463	0.4497	0.4567
600792.SH	2021-11-12	1.4008	1.4116	1.3901	1.4008	1.4224

图 1 数据整理示例

(9) 读取沪深 300 数据 csv 文件 (hs300-2020.1.1-2022.4.22.csv), 观察读取的文件, 并做数据清洗 (形式与 (8) 一致)。(难点提示: 读取的 csv 文件

日期不是 str 类型，建议用 apply 函数 (apply(str))，再变为 DatetimeIndex)

ts_code	trade_date	close	open	high	low	pre_close	pct_chg	vol
399300.SZ	2020-01-02	4152.2408	4121.3487	4172.6555	4121.3487	4096.5821	1.3587	182116772
	2020-01-03	4144.9649	4161.2185	4164.2989	4131.8640	4152.2408	-0.1752	142826244
	2020-01-06	4129.2954	4120.5211	4170.6384	4102.3796	4144.9649	-0.3780	175309953
	2020-01-07	4160.2274	4137.4019	4161.2504	4135.0972	4129.2954	0.7491	139489031
	2020-01-08	4112.3172	4139.6315	4149.8130	4101.9801	4160.2274	-1.1516	167585850
399300.SZ
	2021-11-08	4848.1795	4841.1527	4866.0822	4827.6892	4842.3458	0.1205	137398264
	2021-11-09	4846.7444	4858.2525	4870.7905	4818.5386	4848.1795	-0.0296	118375526
	2021-11-10	4821.1925	4832.8006	4832.8006	4753.9959	4846.7444	-0.5272	135118179
	2021-11-11	4898.6529	4814.7655	4898.6529	4809.1109	4821.1925	1.6067	160484863
	2021-11-12	4888.3749	4902.9289	4909.0839	4875.3492	4898.6529	-0.2098	132662969

图 2 沪深 300 数据整理示例

(10) 观察 (8)、(9) 生成的两个 DataFrame，将两个 DataFrame 的列名调整一致（列名、列名顺序一致）（提示：可以用 `df = df[[列名 1, 列名 2...]]`）

(11) 将整理好的四支股票行情数据与沪深 300 行情数据合并为一个 DataFrame

(12) 将合并后的数据保存为 HDF5 格式文件

(13) 使用数据透视表，即 pivot/pivot_table 的方法，考察 DataFrame 中的收盘价 ‘close’ 并转变为 Index 为日期，columns 为股票代码的形式（提示：在金融数据处理的课件中）

ts_code	000538.SZ	000807.SZ	002053.SZ	399300.SZ	600792.SH
trade_date					
2020-01-02	39.58	2.32	3.24	4152.24	1.55
2020-01-03	39.49	2.30	3.28	4144.96	1.54
2020-01-06	39.03	2.35	3.30	4129.30	1.55
2020-01-07	39.25	2.39	3.33	4160.23	1.69
2020-01-08	38.79	2.28	3.26	4112.32	1.61

图 3 pct_chg 数据整理示例

(14) 以折线图的方式表现四支股票与 hs300 的收盘价 ‘close’ 走势，用子图的方式表示，图的标题设置为 Close Price

(15) 计算或选取五个资产每日收益（离散），并用直方图表示；（提示：先将 (11) 合并的 DataFrame 做 reset_index，选取 pct_chg，操作与 (13) 类似）并用直方图表示

ts_code	000538.SZ	000807.SZ	002053.SZ	399300.SZ	600792.SH
trade_date					
2020-01-02	-0.134250	1.945100	1.105800	1.3587	-0.571000
2020-01-03	-0.234950	-0.953075	1.226825	-0.1752	-0.284000
2020-01-06	-1.167300	2.115975	0.541500	-0.3780	0.569600
2020-01-07	0.567725	1.699925	0.803375	0.7491	8.858750
2020-01-08	-1.163025	-4.637300	-2.129550	-1.1516	-4.721775

图 4 pct_chg 数据整理示例

(16) 计算五个资产离散收益之间的相关性，并用热力图表示

(17) 计算累积收益率（连续），并做出连续收益的折线图，用子图形式表示

其它要求：

- (1) 实验步骤和分析尽可能详细；
- (2) 实验报告正文中写步骤和结论，代码以附件的形式附在报告后；
- (3) 报告整洁，格式规范（正文：仿宋小四；标题：宋体三号；图、表均有标题，黑体五号；注明数据来源，黑体五号）；
- (4) 独立完成，发现抄袭情况不计分；
- (5) 完成后，将**实验报告**和**代码文件**（.ipynb 文件）一起发送至邮箱 21190066@sbs.edu.cn

二. 实验设备、软件

（计算机硬件、系统版本、软件版本）

硬件：

处理器 AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz
机带 RAM 16.0 GB (15.4 GB 可用)
设备 ID 60D9792D-1943-43D8-A88C-1CA51074B34C
产品 ID 00342-35885-54213-AA0EM
系统类型 64 位操作系统，基于 x64 的处理器

软件：

Anaconda 3 - Jupyter notebook - Python3

三. 实验步骤

实验前准备：

第一步打开 Anaconda Navigator, 运行 jupyter notebook。

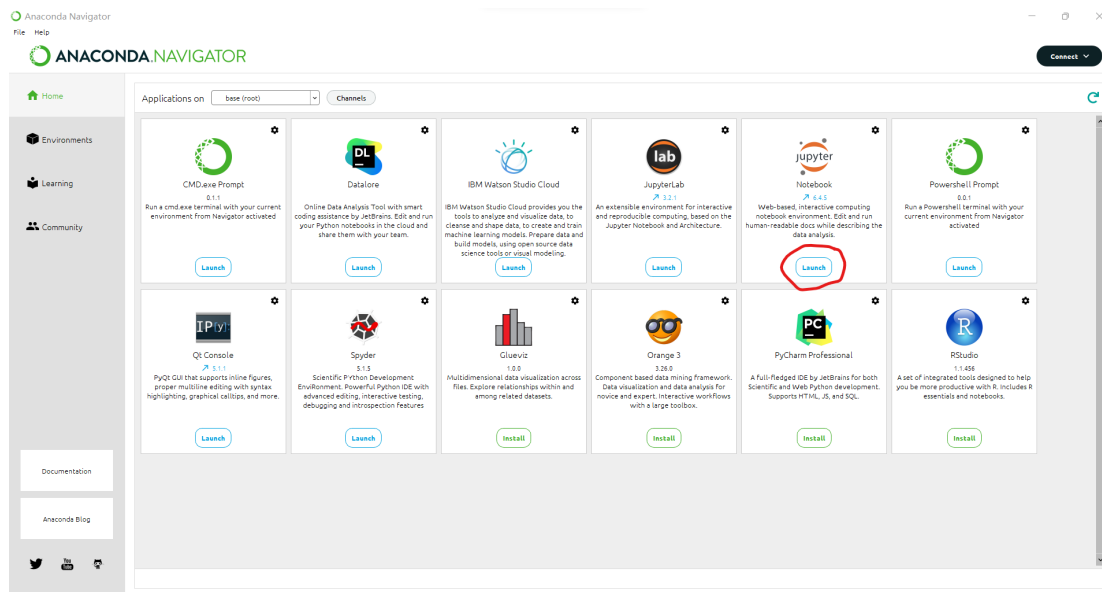


图1 Anaconda Navigator

第二步打开 cmd 命令窗口输入 pip install tushare 安装 tushare。

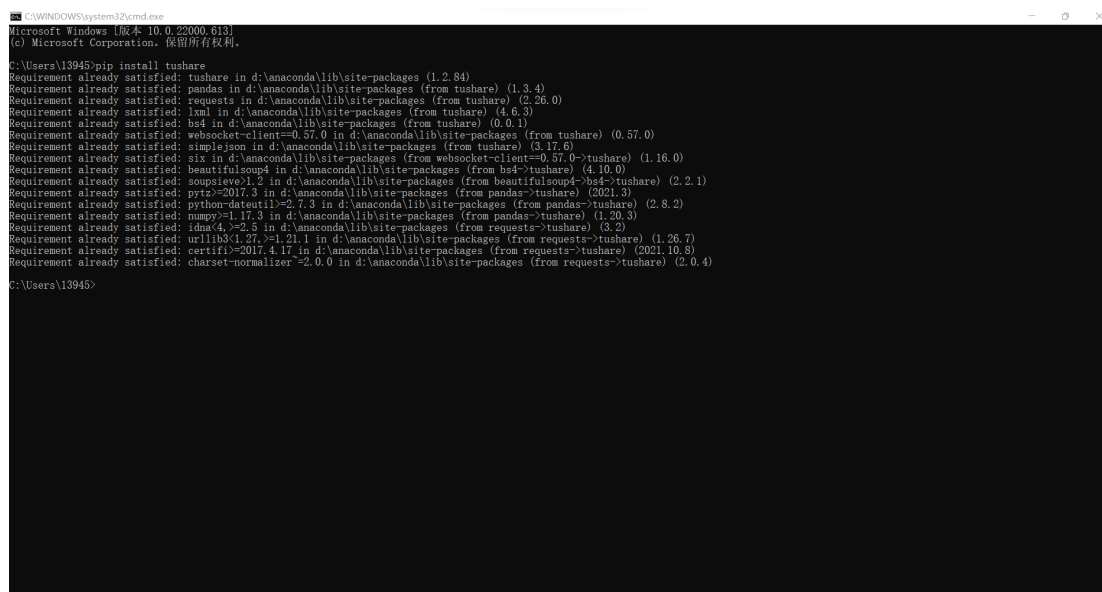


图2 cmd 命令窗口

第三步在 Python 中导入需要的数据库，包括 Tushare，pro_api，matplot，seaborn，pandas，numpy。

```
In [1]: import tushare as ts

In [3]: pro = ts.pro_api()

In [11]: %matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib as mpl
import warnings; warnings.simplefilter('ignore')

In [12]: import pandas as pd
import numpy as np
```

图 3 导入数据库

实验过程：

(1) 通过 Tushare 数据接口 (Pro 版) 获取股票列表中上市公司的列表 (stock_basic 接口), 列表中至少需要包含以下信息: 股票名称 (name), 地区 (area), 行业 (industry), 交易所 (exchange), 股票代码 (ts_code)
第一步: 通过 Tushare 的 stock_basic 接口获取股票列表中上市公司的列表^[1], 命名为 data。

```
In [4]: data = pro.stock_basic(exchange='', list_status='L', fields='ts_code,exchange,name,area,industry')
In [5]: data
Out[5]:
```

	ts_code	name	area	industry	exchange
0	000001.SZ	平安银行	深圳	银行	SZSE
1	000002.SZ	万科A	深圳	全国地产	SZSE
2	000004.SZ	国华网安	深圳	软件服务	SZSE
3	000005.SZ	ST星源	深圳	环境保护	SZSE
4	000006.SZ	深振业A	深圳	区域地产	SZSE
...
4794	871857.BJ	泓禧科技	None	None	BSE
4795	871981.BJ	晶鑫科技	None	None	BSE
4796	872925.BJ	锦好医疗	None	None	BSE
4797	873169.BJ	七丰精工	None	None	BSE
4798	689009.SH	九号公司-WD	北京	摩托车	SSE

图 4 上市公司的列表

(2) 从上市公司列表中筛选出同学自己家乡省份/市的股票列表 (提示: 用 groupby), 并从中任意选取五支股票
第一步: 使用 groupby 函数, 筛选条件为 area, 限定区域为 “江西”。

```
In [6]: grouped = data.groupby('area')
```

```
In [10]: grouped.get_group('江西')
```

```
Out[10]:
```

	ts_code	name	area	industry	exchange
75	000404.SZ	长虹华意	江西	家用电器	SZSE
137	000550.SZ	江铃汽车	江西	汽车整车	SZSE
208	000650.SZ	仁和药业	江西	中成药	SZSE
300	000789.SZ	万年青	江西	水泥	SZSE
325	000820.SZ	*ST节能	江西	环境保护	SZSE
...
4340	688057.SH	金达莱	江西	环境保护	SSE
4469	688223.SH	晶科能源	江西	电气设备	SSE
4613	688560.SH	明冠新材	江西	电气设备	SSE
4617	688567.SH	孚能科技	江西	电气设备	SSE
4705	688786.SH	悦安新材	江西	小金属	SSE

68 rows × 5 columns

图 5 江西省的股票列表

第二步：从江西省的股票列表中任意选取 5 支股票，使用 get_group 函数。

```
In [25]: grouped.get_group('江西').sample(n=5, random_state=None)
```

```
Out[25]:
```

	ts_code	name	area	industry	exchange
4284	605399.SH	晨光新材	江西	化工原料	SSE
442	000990.SZ	诚志股份	江西	化工原料	SZSE
932	002460.SZ	赣锋锂业	江西	小金属	SZSE
2105	300636.SZ	同和药业	江西	化学制药	SZSE
1770	300294.SZ	博雅生物	江西	生物制药	SZSE

图 6 江西省的任意 5 支股票

(3) 获取 2022 年 04 月 22 日所有上市股票的基本面数据，需包括动态市盈率 (pe)、每股收益 (eps)、市净率 (pb)、地域 (area)、股票代码 (ts_code)、股票名称 (name)、行业 (industry) (提示: bak_basic 接口, 参考 tushare 网站基础数据中的备用列表)

第一步：使用 Tushare 的 bak_basic 接口获取 2022 年 04 月 22 日所有上市股票的基本面数据^[2]，命名为 df_basic。

```
In [19]: df_basic = pro.bak_basic(trade_date='20220422', fields='area,ts_code,name,industry,pe,eps,pb')
```

```
In [20]: df_basic
```

```
Out[20]:
```

	ts_code	name	industry	area	pe	eps	pb
0	301288.SZ	N清研	环境保护	深圳	58.08	0.657	5.12
1	301187.SZ	N欧圣	家用电器	江苏	44.42	0.635	3.72
2	301163.SZ	C宏德	机械基件	江苏	41.15	0.805	2.59
3	301212.SZ	C联盛	化工原料	浙江	67.98	0.820	5.22
4	300952.SZ	恒辉安防	纺织	江苏	30.78	0.639	3.08
...
4711	603191.SH	望变电气	电气设备	重庆	0.00	0.000	0.00
4712	603097.SH	江苏华辰	电气设备	江苏	0.00	0.000	0.00
4713	301259.SZ	艾布鲁	环境保护	湖南	0.00	0.000	0.00
4714	301162.SZ	国能日新	软件服务	北京	0.00	0.000	0.00
4715	002260.SZ	*ST德奥	家用电器	广东	0.00	-0.030	0.00

4716 rows × 7 columns

图 7 20220422 上市股票基本面数据

(4) 删除 DataFrame 中 industry 列

第一步：使用 del 函数删除 df_basic 中的 industry 列。

```
In [21]: del df_basic['industry']
```

```
In [22]: df_basic
```

```
Out[22]:
```

	ts_code	name	area	pe	eps	pb
0	301288.SZ	N清研	深圳	58.08	0.657	5.12
1	301187.SZ	N欧圣	江苏	44.42	0.635	3.72
2	301163.SZ	C宏德	江苏	41.15	0.805	2.59
3	301212.SZ	C联盛	浙江	67.98	0.820	5.22
4	300952.SZ	恒辉安防	江苏	30.78	0.639	3.08
...
4711	603191.SH	望变电气	重庆	0.00	0.000	0.00
4712	603097.SH	江苏华辰	江苏	0.00	0.000	0.00
4713	301259.SZ	艾布鲁	湖南	0.00	0.000	0.00
4714	301162.SZ	国能日新	北京	0.00	0.000	0.00
4715	002260.SZ	*ST德奥	广东	0.00	-0.030	0.00

4716 rows × 6 columns

图 8 删除 df_basic 中的 industry 列

(5) 将股票代码设置为 index，从获取的 DataFrame 中选取并查看所选取五支股票的 pe、eps、pb 数据（提示：用 loc）

第一步：使用 set_index 函数，将股票代码设置为 index。

```
In [23]: df_basic.set_index('ts_code', inplace=True)
```


图 9 股票代码设置为 index

第二步：使用 loc 函数，选取并查看所选取五支股票的 pe、eps、pb 数据。

```
In [28]: data1 = df_basic.loc[['605399.SH', '000990.SZ', '002460.SZ', '300636.SZ', '300294.SZ']]
data1
```

```
Out[28]:
```

	pe	eps	pb
ts_code			
605399.SH	21.03	1.63	6.00
000990.SZ	13.27	0.83	0.76
002460.SZ	27.88	3.73	6.66
300636.SZ	48.09	0.32	4.72
300294.SZ	38.46	0.79	1.94

图 10 五支股票的 pe、eps、pb 数据

(6) 删除上述五只股票中 eps 值最小的一支股票数据（剩四支），形成新的 DataFrame

第一步：使用 drop 函数，删除上述五只股票中 eps 值最小的一支股票数据，命名新的 DataFrame 为 df_clear。

```
In [33]: df_clear = data1.drop('300636.SZ', axis=0)
```

```
In [34]: df_clear
```

```
Out[34]:
```

	pe	eps	pb
ts_code			
605399.SH	21.03	1.63	6.00
000990.SZ	13.27	0.83	0.76
002460.SZ	27.88	3.73	6.66
300294.SZ	38.46	0.79	1.94

图 11 删除 eps 值最小的股票数据后剩余的四支股票数据

(7) 从任意免费金融数据接口（建议 Tushare Pro 版）获取所选出的四支股票 2020 年 1 月 1 日到 2022 年 04 月 22 日的前复权的行情数据；（提示：通用行情接口，pro_bar）

第一步：使用 Tushare 的通用行情接口 pro_bar，获取所选出的四支股票 2020 年 1 月 1 日到 2022 年 04 月 22 日的前复权的行情数据^[3]，命名为 data2。

```
In [38]: data2 = ts.pro_bar(ts_code='605399.SH,000990.SZ,002460.SZ,300294.SZ', adj='qfq', freq='d', start_date='20200101', end_date='20220422')
data2.head()
```

```
Out[38]:
```

	ts_code	trade_date	open	high	low	close	pre_close	change	pct_chg	vol	amount
0	000990.SZ	20200102	14.7700	14.7700	14.3700	14.7000	14.8200	-0.1200	-0.8097	112419.08	164325.599
1	000990.SZ	20200102	45.0958	45.0958	43.8746	44.8821	45.2485	-0.3664	-0.8098	112419.08	164325.599
2	000990.SZ	20200102	22.5290	22.5290	21.9189	22.4222	22.6053	-0.1831	-0.8100	112419.08	164325.599
3	002460.SZ	20200102	35.0500	35.7800	34.3300	35.4600	34.8300	0.6300	1.8088	441532.60	1554001.262
4	002460.SZ	20200102	107.0148	109.2437	104.8165	108.2667	108.3431	1.9236	1.8089	441532.60	1554001.262

图 12 四支股票的前复权行情数据

(8) 清洗、整理所获取的数据，如下图形式（包括将股票代码和交易时间设置

为双重索引 Multiindex，将 index 转变为 DatetimeIndex，时间排序等)

第一步：使用 to_datetime 函数，将 trade_date 转变为 Datetime。

```
In [39]: data2.trade_date = pd.to_datetime(data2.trade_date)
```

图 13 trade_date 转变为 Datetime

第二步：使用 sort_values 函数，对 trade_date 进行时间排序。

```
In [40]: data2.sort_values('trade_date', ascending=True, inplace=True)
```

图 14 trade_date 时间排序

第三步：使用 set_index 函数，将股票代码和交易时间设置为双重索引 Multiindex。

```
In [41]: data2.set_index(['ts_code', 'trade_date'], inplace=True)
data2.head()

Out[41]:
```

	ts_code	trade_date	open	high	low	close	pre_close	change	pct_chg	vol	amount
000990.SZ	2020-01-02	14.7700	14.7700	14.3700	14.7000	14.8200	-0.1200	-0.8097	-0.8098	112419.08	164325.599
	2020-01-02	45.0958	45.0958	43.8746	44.8821	45.2485	-0.3664	-0.8098	-0.8098	112419.08	164325.599
	2020-01-02	22.5290	22.5290	21.9189	22.4222	22.6053	-0.1831	-0.8100	-0.8100	112419.08	164325.599
002460.SZ	2020-01-02	35.0500	35.7800	34.3300	35.4600	34.8300	0.6300	1.8088	4.1532	441532.60	1554001.262
	2020-01-02	107.0148	109.2437	104.8165	108.2667	106.3431	1.9236	1.8089	4.1532	441532.60	1554001.262

图 15 双重索引股票代码和交易时间

(9) 读取沪深 300 数据 csv 文件 (hs300-2020.1.1-2022.4.22.csv)，观察读取的文件，并做数据清洗（形式与 (8) 一致）。（难点提示：读取的 csv 文件日期不是 str 类型，建议用 apply 函数 (apply(str))，再变为 DatetimeIndex）

第一步：使用 read_csv 函数，读取沪深 300 数据 csv 文件 (hs300-2020.1.1-2022.4.22.csv)^[4]，命名为 ds_hs300。

```
In [84]: df_hs300 = pd.read_csv('C:/Users/13945/Desktop/hs300-2020.1.1-2022.4.22.csv')
df_hs300.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 558 entries, 0 to 557
Data columns (total 12 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   Unnamed: 0   558 non-null    int64
1   ts_code      558 non-null    object
2   trade_date   558 non-null    int64
3   close        558 non-null    float64
4   open         558 non-null    float64
5   high         558 non-null    float64
6   low          558 non-null    float64
7   pre_close    558 non-null    float64
8   change       558 non-null    float64
9   pct_chg      558 non-null    float64
10  vol          558 non-null    float64
11  amount       558 non-null    float64
dtypes: float64(9), int64(2), object(1)
memory usage: 52.4+ KB
```

图 16 沪深 300 数据

第二步：使用 `apply(str)` 函数，将读取的 csv 文件日期转化为 `str` 类型，再使用 `to_datetime` 函数，将 `trade_date` 转变为 `Datetime`。

```
In [86]: df_hs300['trade_date'] = df_hs300['trade_date'].apply(str)

In [87]: df_hs300.trade_date = pd.to_datetime(df_hs300.trade_date)
```

图 17 转化后的 hs300 数据

第三步：使用 `del` 函数，删去多余的 column，再使用 `sort_values` 函数，对 `trade_date` 进行时间排序。

```
In [88]: del df_hs300['Unnamed: 0']
df_hs300.head()

Out[88]:
```

	ts_code	trade_date	close	open	high	low	pre_close	change	pct_chg	vol	amount
0	399300.SZ	2022-04-22	4013.2498	3967.2016	4037.0840	3953.7217	3995.8300	17.4198	0.4359	122101811.0	206371766.7
1	399300.SZ	2022-04-21	3995.8300	4048.7338	4086.4393	3978.3669	4070.7889	-74.9589	-1.8414	132783828.0	227026679.6
2	399300.SZ	2022-04-20	4070.7889	4130.5427	4136.1661	4060.5044	4134.9017	-64.1128	-1.5505	117331658.0	216536635.6
3	399300.SZ	2022-04-19	4134.9017	4161.9328	4186.9389	4115.4184	4166.3844	-31.4827	-0.7556	110217471.0	208570579.1
4	399300.SZ	2022-04-18	4166.3844	4152.9306	4171.7427	4119.9941	4188.7472	-22.3628	-0.5339	117504610.0	212855328.1

```
In [89]: df_hs300.sort_values('trade_date', ascending=True, inplace=True)
```

图 18 对 hs300 数据进行清洗

第四步：使用 `set_index` 函数，将股票代码和交易时间设置为双重索引 `Multiindex`。

```
In [90]: df_hs300.set_index(['ts_code', 'trade_date'], inplace=True)
df_hs300

Out[90]:
```

	ts_code	trade_date	close	open	high	low	pre_close	change	pct_chg	vol	amount
399300.SZ		2020-01-02	4152.2408	4121.3487	4172.6555	4121.3487	4096.5821	55.6587	1.3587	182116772.0	270105532.0
		2020-01-03	4144.9649	4161.2185	4164.2989	4131.8640	4152.2408	-7.2759	-0.1752	142826244.0	215216288.3
		2020-01-06	4129.2954	4120.5211	4170.6384	4102.3796	4144.9649	-15.6695	-0.3780	175309953.0	250182071.2
		2020-01-07	4160.2274	4137.4019	4161.2504	4135.0972	4129.2954	30.9320	0.7491	139489031.0	196389059.7
		2020-01-08	4112.3172	4139.6315	4149.8130	4101.9801	4160.2274	-47.9102	-1.1516	167585850.0	212406263.8
...		
2022-04-18		2022-04-18	4166.3844	4152.9306	4171.7427	4119.9941	4188.7472	-22.3628	-0.5339	117504610.0	212855328.1
		2022-04-19	4134.9017	4161.9328	4186.9389	4115.4184	4166.3844	-31.4827	-0.7556	110217471.0	208570579.1
		2022-04-20	4070.7889	4130.5427	4136.1661	4060.5044	4134.9017	-64.1128	-1.5505	117331658.0	216536635.6
		2022-04-21	3995.8300	4048.7338	4086.4393	3978.3669	4070.7889	-74.9589	-1.8414	132783828.0	227026679.6
		2022-04-22	4013.2498	3967.2016	4037.0840	3953.7217	3995.8300	17.4198	0.4359	122101811.0	206371766.7

558 rows × 9 columns

图 19 hs300 双重索引股票代码和交易时间

(10) 观察 (8)、(9) 生成的两个 DataFrame，将两个 DataFrame 的列名调整一致（列名、列名顺序一致）（提示：可以用 `df = df[['列名 1', '列名 2...']]`）

第一步：将 (8) (9) 生成的两个 DataFrame 的列名调整一致。

```
In [91]: df_hs300 = df_hs300[['open','high','low','close','pre_close','change','pct_chg','vol','amount']]
```

```
In [92]: df_hs300
```

```
Out[92]:
```

		open	high	low	close	pre_close	change	pct_chg	vol	amount
399300.SZ	2020-01-02	4121.3487	4172.6555	4121.3487	4152.2408	4096.5821	55.6587	1.3587	182116772.0	270105532.0
	2020-01-03	4161.2185	4164.2989	4131.8640	4144.9649	4152.2408	-7.2759	-0.1752	142826244.0	215216288.3
	2020-01-06	4120.5211	4170.6384	4102.3796	4129.2954	4144.9649	-15.6695	-0.3780	175309953.0	250182071.2
	2020-01-07	4137.4019	4161.2504	4135.0972	4160.2274	4129.2954	30.9320	0.7491	139489031.0	196389059.7
	2020-01-08	4139.6315	4149.8130	4101.9801	4112.3172	4160.2274	-47.9102	-1.1516	167585850.0	212406263.8

	2022-04-18	4152.9306	4171.7427	4119.9941	4166.3844	4188.7472	-22.3628	-0.5339	117504610.0	212855328.1
	2022-04-19	4161.9328	4186.9389	4115.4184	4134.9017	4166.3844	-31.4827	-0.7556	110217471.0	208570579.1
	2022-04-20	4130.5427	4136.1661	4060.5044	4070.7889	4134.9017	-64.1128	-1.5505	117331658.0	216536635.6
	2022-04-21	4048.7338	4086.4393	3978.3669	3995.8300	4070.7889	-74.9589	-1.8414	132783828.0	227026679.6
	2022-04-22	3967.2016	4037.0840	3953.7217	4013.2498	3995.8300	17.4198	0.4359	122101811.0	206371766.7

558 rows × 9 columns

图 20 hs300 的列名与 data2 一致

(11) 将整理好的四支股票行情数据与沪深 300 行情数据合并为一个 DataFrame

第一步：使用 concat 函数，将整理好的四支股票行情数据与沪深 300 行情数据合并为一个 DataFrame，命名为 df_all。

```
In [93]: df_all = pd.concat([df_hs300,data2],axis=0)
```

```
In [94]: df_all
```

```
Out[94]:
```

		open	high	low	close	pre_close	change	pct_chg	vol	amount
399300.SZ	2020-01-02	4121.3487	4172.6555	4121.3487	4152.2408	4096.5821	55.6587	1.3587	1.821168e+08	2.701055e+08
	2020-01-03	4161.2185	4164.2989	4131.8640	4144.9649	4152.2408	-7.2759	-0.1752	1.428262e+08	2.152163e+08
	2020-01-06	4120.5211	4170.6384	4102.3796	4129.2954	4144.9649	-15.6695	-0.3780	1.753100e+08	2.501821e+08
	2020-01-07	4137.4019	4161.2504	4135.0972	4160.2274	4129.2954	30.9320	0.7491	1.394890e+08	1.963891e+08
	2020-01-08	4139.6315	4149.8130	4101.9801	4112.3172	4160.2274	-47.9102	-1.1516	1.675858e+08	2.124063e+08

	2022-04-22	44.6600	47.4000	44.0500	45.4800	44.8500	0.6300	1.4047	4.049621e+04	1.855794e+05
	2022-04-22	137.2579	145.6790	135.3832	139.7781	137.8419	1.9362	1.4047	4.049621e+04	1.855794e+05
	2022-04-22	157.4011	159.8401	154.3332	155.5757	159.5947	-4.0190	-2.5183	1.945127e+05	1.989524e+06
	2022-04-22	33.2825	33.7982	32.6338	32.8965	33.7463	-0.8498	-2.5182	1.945127e+05	1.989524e+06

图 21 hs300 和 data2 合并后的 DataFrame

(12) 将合并后的数据保存为 HDF5 格式文件

第一步：导入 h5py 数据库，将合并后的数据 df_all 保存为 HDF5 格式文件。

```

In [97]: import h5py

In [100]: f = h5py.File('HDF5_FILE.h5', 'w')

In [102]: f['df_all'] = df_all

In [103]: f.close()

```

图 22 df_all 保存为 HDF5 格式文件

(13) 使用数据透视表，即 pivot/pivot_table 的方法，考察 DataFrame 中的收盘价 ‘close’ 并转变为 Index 为日期，columns 为股票代码的形式（提示：在金融数据处理的课件中）

第一步：选取 df_all 的收盘价 ‘close’，并使用 reset_index 函数重置 df_all 的 index，命名为 close_price。

```

In [108]: df_all.close

Out[108]: ts_code    trade_date    close
          399300.SZ  2020-01-02    4152.2408
          399300.SZ  2020-01-03    4144.9649
          399300.SZ  2020-01-06    4129.2954
          399300.SZ  2020-01-07    4160.2274
          399300.SZ  2020-01-08    4112.3172
          ...
          605399.SH  2022-04-22    45.4800
          605399.SH  2022-04-22    139.7781
          002460.SZ  2022-04-22    155.5757
          002460.SZ  2022-04-22    32.8965
          605399.SH  2022-04-22    14.7518
Name: close, Length: 8484, dtype: float64

In [109]: close_price = df_all[['close']].reset_index()

In [110]: close_price.tail()

Out[110]:
   ts_code trade_date  close
8479  605399.SH  2022-04-22  45.4800
8480  605399.SH  2022-04-22  139.7781
8481  002460.SZ  2022-04-22  155.5757
8482  002460.SZ  2022-04-22  32.8965
8483  605399.SH  2022-04-22  14.7518

```

图 23 对 df_all 的收盘价 ‘close’ 进行数据清洗

第二步：使用数据透视表，即 pivot_table 的方法，将 close_price 中的收盘价 ‘close’ 并转变为 Index 为日期，columns 为股票代码的形式，命名为 daily_close。

```
In [112]: daily_close = close_price.pivot_table(index='trade_date', columns='ts_code', values='close', aggfunc='sum')
daily_close
```

```
Out[112]:
```

	ts_code	000990.SZ	002460.SZ	300294.SZ	399300.SZ	605399.SH
trade_date						
2020-01-02	82.0043	197.8146	175.2773	4152.2408	NaN	
2020-01-03	81.6696	202.9468	181.7483	4144.9649	NaN	
2020-01-06	83.2874	223.1410	174.5521	4129.2954	NaN	
2020-01-07	83.2874	223.0294	174.9983	4160.2274	NaN	
2020-01-08	82.7854	226.8228	175.1657	4112.3172	NaN	
...	
2022-04-18	72.9603	683.8100	158.6738	4166.3844	257.0219	
2022-04-19	73.0196	677.1071	157.4282	4134.9017	269.0633	
2022-04-20	67.0286	649.5245	157.0722	4070.7889	270.9022	
2022-04-21	63.5881	617.1373	156.5384	3995.8300	266.0381	
2022-04-22	63.3509	601.5962	153.6318	4013.2498	269.7751	

558 rows x 5 columns

图 24 数据透视表

(14) 以折线图的方式表现四支股票与 hs300 的收盘价 ‘close’ 走势，用子图的方式表示，图的标题设置为 Close Price

第一步:使用 plot 函数,以折线图的方式表现四支股票与 hs300 的收盘价 ‘close’ 走势，用子图的方式表示，图的标题设置为 Close Price。

```
In [114]: daily_close.plot(subplots=True, figsize=(10, 8), title='Close Price')
```

```
Out[114]: array([<AxesSubplot: xlabel='trade_date'>,
<AxesSubplot: xlabel='trade_date'>,
<AxesSubplot: xlabel='trade_date'>,
<AxesSubplot: xlabel='trade_date'>,
<AxesSubplot: xlabel='trade_date'>], dtype=object)
```



图 25 五支股票的收盘价 ‘close’ 走势——折线图

(15) 计算或选取五个资产每日收益(离散), 并用直方图表示; (提示: 先将(11)合并的 DataFrame 做 reset_index, 选取 pct_chg, 操作与 (13) 类似) 并用直方图表示

第一步: 使用 shift 函数, 计算五个资产每日收益 (离散)。

```
In [115]: daily_close.head()
```

Out[115]:

	ts_code	000990.SZ	002460.SZ	300294.SZ	399300.SZ	605399.SH
trade_date	2020-01-02	82.0043	197.8146	175.2773	4152.2408	NaN
2020-01-03	81.6696	202.9468	181.7483	4144.9649	NaN	
2020-01-06	83.2874	223.1410	174.5521	4129.2954	NaN	
2020-01-07	83.2874	223.0294	174.9983	4160.2274	NaN	
2020-01-08	82.7854	226.8228	175.1657	4112.3172	NaN	

```
In [116]: price_change = daily_close/daily_close.shift(1)-1
price_change.head()
```

Out[116]:

	ts_code	000990.SZ	002460.SZ	300294.SZ	399300.SZ	605399.SH
trade_date	2020-01-02	NaN	NaN	NaN	NaN	NaN
2020-01-03	-0.004081	0.025944	0.036919	-0.001752	NaN	
2020-01-06	0.019809	0.099505	-0.039594	-0.003780	NaN	
2020-01-07	0.000000	-0.000500	0.002556	0.007491	NaN	
2020-01-08	-0.006027	0.017009	0.000957	-0.011516	NaN	

图 26 五个资产每日收益

第二步: 使用 fillna 函数, 对 price_change 进行数据清洗。


```
In [117]: price_change.fillna(0,inplace=True)
price_change.head()
```

```
Out[117]:
```

	ts_code	000990.SZ	002460.SZ	300294.SZ	399300.SZ	605399.SH
	trade_date					
	2020-01-02	0.000000	0.000000	0.000000	0.000000	0.0
	2020-01-03	-0.004081	0.025944	0.036919	-0.001752	0.0
	2020-01-06	0.019809	0.099505	-0.039594	-0.003780	0.0
	2020-01-07	0.000000	-0.000500	0.002556	0.007491	0.0
	2020-01-08	-0.006027	0.017009	0.000957	-0.011516	0.0

图 27 对 price_change 进行数据清洗

第三步：使用 hist 函数，用直方图表示五个资产每日收益（离散）。

```
In [119]: price_change.hist(bins=30,figsize=(10,8))
```

```
Out[119]: array([[<AxesSubplot:title=' center ': '000990.SZ'>],
<AxesSubplot:title=' center ': '002460.SZ'>],
[<AxesSubplot:title=' center ': '300294.SZ'>],
<AxesSubplot:title=' center ': '399300.SZ'>],
[<AxesSubplot:title=' center ': '605399.SH'>], <AxesSubplot:>]],
dtype=object)
```

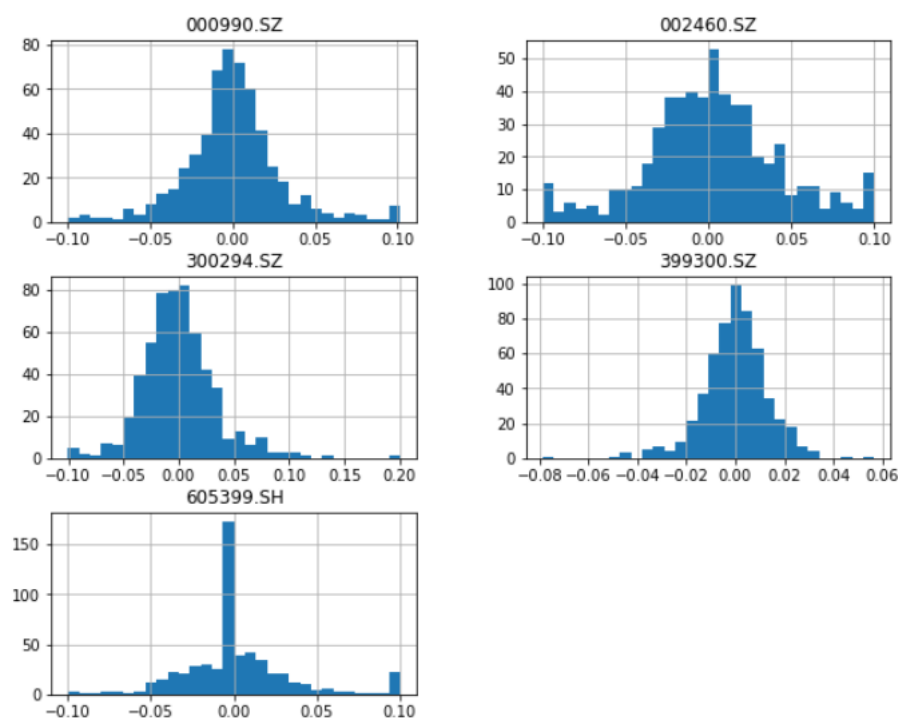


图 28 五个资产每日收益（离散）——直方图

(16) 计算五个资产离散收益之间的相关性，并用热力图表示

第一步：使用 corr 函数，计算五个资产离散收益之间的相关性。


```

In [120]: all_pct_chg=df_all['pct_chg']/100
          all_pct_chg.head()

Out[120]: ts_code    trade_date
          399300.SZ  2020-01-02    0.013587
                   2020-01-03   -0.001752
                   2020-01-06   -0.003780
                   2020-01-07    0.007491
                   2020-01-08   -0.011516
          Name: pct_chg, dtype: float64

In [121]: all_pct_chg.sort_index(ascending=True,inplace=True)
          all_pct_chg.head()

Out[121]: ts_code    trade_date
          000990.SZ  2020-01-02   -0.008097
                   2020-01-02   -0.008098
                   2020-01-02   -0.008100
                   2020-01-03   -0.004082
                   2020-01-03   -0.004081
          Name: pct_chg, dtype: float64

```

图 29.1 计算五个资产离散收益之间的相关性

```

In [122]: df=pd.DataFrame()
          all_pct_chg=pd.DataFrame(all_pct_chg)

In [123]: all_pct_chg

Out[123]:

```

		pct_chg
ts_code	trade_date	
000990.SZ	2020-01-02	-0.008097
	2020-01-02	-0.008098
	2020-01-02	-0.008100
	2020-01-03	-0.004082
	2020-01-03	-0.004081
...
605399.SH	2022-04-21	-0.017955
	2022-04-22	0.014048
	2022-04-22	0.014047
	2022-04-22	0.014047

图 29.2 计算五个资产离散收益之间的相关性

```
In [125]: all_pct_chg1 = all_pct_chg.reset_index()
```

```
In [126]: all_pct_chg1
```

```
Out[126]:
```

	ts_code	trade_date	pct_chg
0	000990.SZ	2020-01-02	-0.008097
1	000990.SZ	2020-01-02	-0.008098
2	000990.SZ	2020-01-02	-0.008100
3	000990.SZ	2020-01-03	-0.004082
4	000990.SZ	2020-01-03	-0.004081
...
8479	605399.SH	2022-04-21	-0.017955
8480	605399.SH	2022-04-22	0.014048
8481	605399.SH	2022-04-22	0.014047
8482	605399.SH	2022-04-22	0.014047
8483	605399.SH	2022-04-22	0.014044

图 29.3 计算五个资产离散收益之间的相关性

```
In [129]: all_pct_chg2 = all_pct_chg1.pivot_table(index='trade_date', columns='ts_code', values='pct_chg', aggfunc='sum')
all_pct_chg2
```

```
Out[129]:
```

ts_code	000990.SZ	002460.SZ	300294.SZ	399300.SZ	605399.SH
trade_date					
2020-01-02	-0.024295	0.054266	0.018252	0.013587	NaN
2020-01-03	-0.012245	0.077834	0.110756	-0.001752	NaN
2020-01-06	0.059426	0.298514	-0.118784	-0.003780	NaN
2020-01-07	0.000000	-0.001500	0.007669	0.007491	NaN
2020-01-08	-0.018081	0.051026	0.002869	-0.011516	NaN
...
2022-04-18	0.046054	0.027248	0.091778	-0.005339	-0.007368
2022-04-19	0.003263	-0.039208	-0.031399	-0.007556	0.187396
2022-04-20	-0.328185	-0.162945	-0.009049	-0.015505	0.027341
2022-04-21	-0.205328	-0.199452	-0.013595	-0.018414	-0.071822
2022-04-22	-0.014905	-0.100731	-0.074269	0.004359	0.056186

558 rows × 5 columns

图 29.4 计算五个资产离散收益之间的相关性

```
In [130]: corrs = all_pct_chg2.corr()
          corrs
```

```
Out[130]:
```

	ts_code	000990.SZ	002460.SZ	300294.SZ	399300.SZ	605399.SH
ts_code						
000990.SZ		1.000000	0.137808	0.160252	0.297006	0.039494
002460.SZ		0.137808	1.000000	0.208982	0.471032	0.190231
300294.SZ		0.160252	0.208982	1.000000	0.380864	0.092759
399300.SZ		0.297006	0.471032	0.380864	1.000000	0.087119
605399.SH		0.039494	0.190231	0.092759	0.087119	1.000000

图 29.5 计算五个资产离散收益之间的相关性

第二步：使用 seaborn 里的 heatmap 绘图，用热力图表示五个资产离散收益之间的相关性。

```
In [131]: import seaborn
          fig = plt.figure(figsize=(8,6))
          seaborn.heatmap(corrs)
```

```
Out[131]: <AxesSubplot:xlabel='ts_code', ylabel='ts_code'>
```

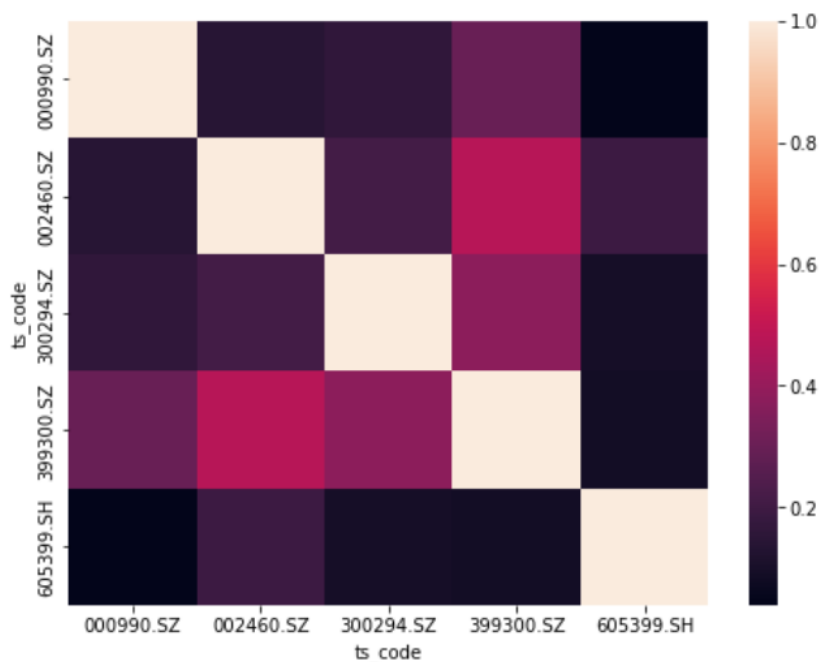


图 30 五个资产离散收益之间的相关性——热力图

(17) 计算累积收益率（连续），并做出连续收益的折线图，用子图形式表示
 第一步：使用 cumprod 函数计算累积收益率，命名为 cumreturn_all。

```
In [132]: cumreturn_all = (1+all_pct_chg2).cumprod()
          cumreturn_all.tail()
```

Out[132]:

ts_code	000990.SZ	002460.SZ	300294.SZ	399300.SZ	605399.SH
trade_date					
2022-04-18	0.041665	0.564434	0.014036	1.017046	2.983628
2022-04-19	0.041801	0.542304	0.013596	1.009361	3.542748
2022-04-20	0.028083	0.453938	0.013473	0.993711	3.639610
2022-04-21	0.022317	0.363399	0.013290	0.975413	3.378206
2022-04-22	0.021984	0.326794	0.012303	0.979665	3.568014

图 31 计算累计收益率

第二步：使用 plot 函数，做出连续收益的折线图，用子图形式表示。

```
In [136]: cumreturn_all.plot(subplots=True,figsize=(10,8))
```

```
Out[136]: array([<AxesSubplot:xlabel='trade_date'>,
                  <AxesSubplot:xlabel='trade_date'>,
                  <AxesSubplot:xlabel='trade_date'>,
                  <AxesSubplot:xlabel='trade_date'>,
                  <AxesSubplot:xlabel='trade_date'>], dtype=object)
```

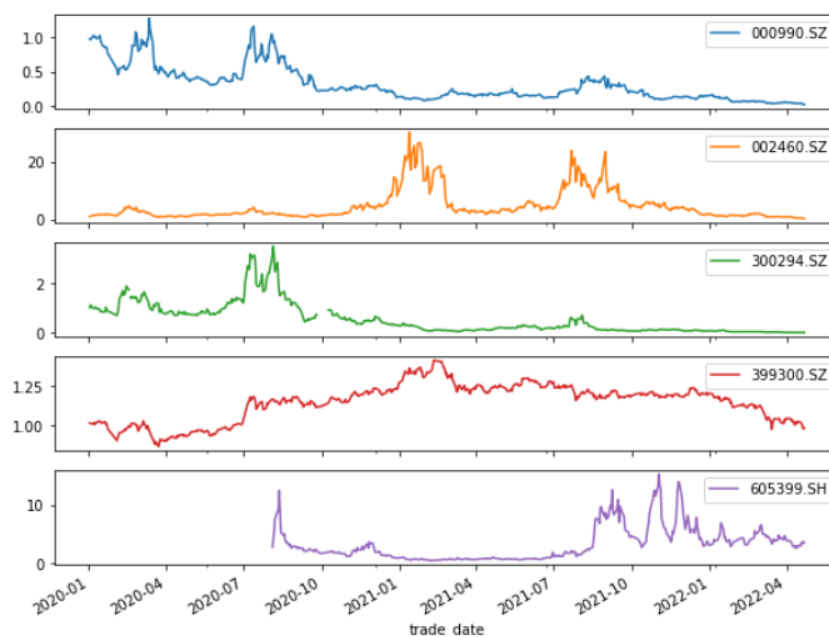


图 32 连续收益的折线图

数据来源：

- [1]通过 Tushare 的 stock_basic 接口获取股票列表中上市公司的列表
- [2]使用 Tushare 的 bak_basic 接口获取 2022 年 04 月 22 日所有上市股票的基本面数据
- [3]使用 Tushare 的通用行情接口 pro_bar，获取所选出的四支股票 2020 年 1 月 1 日到 2022 年 04 月 22 日的前复权的行情数据
- [4]读取沪深 300 数据 csv 文件 (hs300-2020.1.1-2022.4.22.csv)

四. 实验结果及总结

通过从网络数据接口获取金融数据，并对获取的数据进行预处理，使我们掌握使用 Python 对金融数据初步分析的基本方法。

通过本次实验，让我初步接触和应用 Python，了解到金融数据处理分析的基本方法，掌握了对缺失值填充和处理方法，了解到大数据以及大数据的广泛应用性。

*五. 对本课程的建议和意见

无，但是刘伟荣老师真的很帅！

附件：



(代码)

Python金融数据分析.ipynb