PROGRAMMING FOR BUSINESS COMPUTING
商管程式設計

Applications in finance

Hsin-Min Lu
盧信銘
台大資管系

Objectives

- To understand the typical process to analyze real world financial datasets.
- To understand how to leverage Python to analyze financial datasets.
  - Data preprocessing: read write files, read write csv files
  - Regression: estimate statistical models
  - Processing many stocks: looping
  - Visualize result: matplotlib

股票市場

- 股票市場分為初級市場與次級市場
  - 初級市場：又稱為「發行市場」，是指企業提供新的證券銷售給社會大眾的市場，又稱為「第一市場」。
  - 次級市場：又稱為「流通市場」，是指社會大眾購買新證券之後，這些證券後續買賣的市場。
- 我們平常看到的股價與交易資訊大都來自次級市場。☼

交易所

- 臺灣證券交易所股份有限公司 (Taiwan Stock Exchange Corporation, TSEC)
  - 成立於 1961 年 10 月 3 日，1962 年 2 月 9 日開業，為臺灣證券集中市場。
  - 交易所為「股份有限公司」，為公司制
- 交易時間：星期一至星期五每日 09:00~13:30。
  - 休假停市日除經特別公告外，與金融業的例行假日相同。
  - 臺灣地區遇天然災害時，證券集中市場之休市視當地縣市首長宣佈公教機關是否上班為準 (例如颱風來襲，臺北市若宣佈停止上班，則臺灣證交所休市)。☼

市場概況

單位：10億元

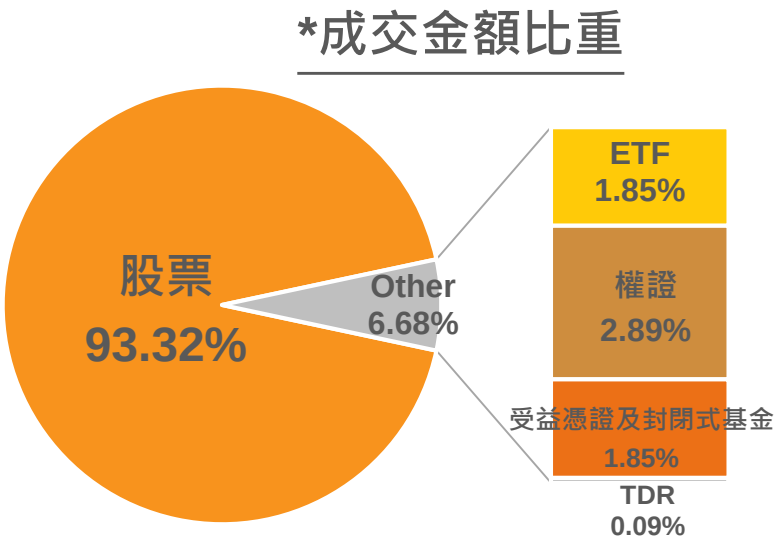
公開發行公司	上市公司	上櫃公司	興櫃公司
家數	854	685	284
市值	26,891.50	2,680.56	893.02

2014年12月，資料來源：台灣證券交易所、櫃檯買賣中心

台灣股市與國際市場比較

項目	統計值	全球排名
•2014年底上市公司家數 (國內外第一上市)	854	第15名
•TDR家數	26	
2014年底總市值 (新台幣10億元)	26,891.5	第18名
2014年總成交金額* (新台幣10億元)	21,898.5	第16名
2014年成交金額周轉率	82.6%	第13名
2014年底ETF掛牌數	25	第23名

資料來源：證券暨期貨市場重要指標、WFE Statistics
註：全球排名係依WFE全體會員共56家交易所計算



註：成交金額比重係依2014年資料計算

股票歷史日資料

證券代碼	簡稱	TSE 產業別	年月日	開盤價(元)	最高價(元)	最低價(元)	收盤價(元)
COID	Name	IND1	MDATE	OPEN	HIGH	LOW	CLOSE
1101	台泥	1	1/3/2005	10.4	10.8	10.4	10.65
1102	亞泥	1	1/3/2005	7.81	7.95	7.78	7.92
1103	嘉泥	1	1/3/2005	11.21	11.5	11.14	11.36
1104	環泥	1	1/3/2005	6.24	6.34	6.19	6.34
1108	幸福	1	1/3/2005	6.45	6.66	6.42	6.62
1109	信大	1	1/3/2005	6.87	6.96	6.84	6.9
1101	台泥	1	1/4/2005	10.65	10.65	10.5	10.5
1102	亞泥	1	1/4/2005	7.88	7.88	7.71	7.74

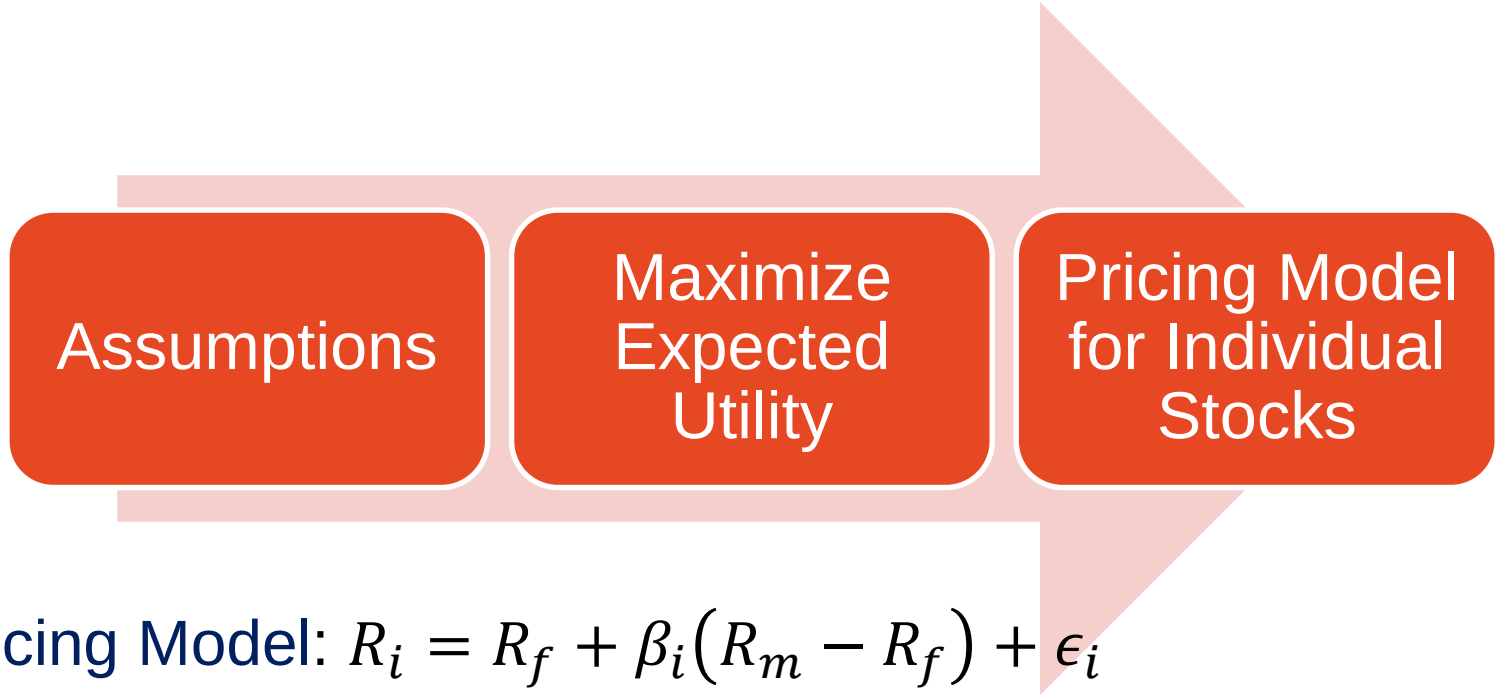
How do we Analyze Stock Return Data?

- We are going to adopt Capital Asset Pricing Model (CAPM) to analyze stock return data.
- CAPM was invented by Jack Treynor (1961), William F. Sharpe (1964), John Lintner (1965) and Jan Mossin (1966) independently.
- Sharpe, Markowitz and Merton Miller jointly received the 1990 Nobel Memorial Prize in Economics for this contribution to the field of financial economics.
- Standard textbook approach.

CAPM in Five Minutes

- Assumptions:
- Investors are rational and risk-averse.
- Investors aim to maximize economic utilities.
- Investors broadly diversified across a range of investments.
- Investors are price takers.
- Investors can lend and borrow unlimited amounts under the risk free rate of interest.
- Investors can trade without transaction or taxation costs.
- All information is available at the same time to all investors.
- All investors have homogeneous expectations.

CAPM in Five Minutes (Cont'd.)



- Pricing Model: $R_i = R_f + \beta_i(R_m - R_f) + \epsilon_i$
- R_i: Return of stock i
- R_m: Market return
- R_f: Risk free rate
- ϵ_i: Noise

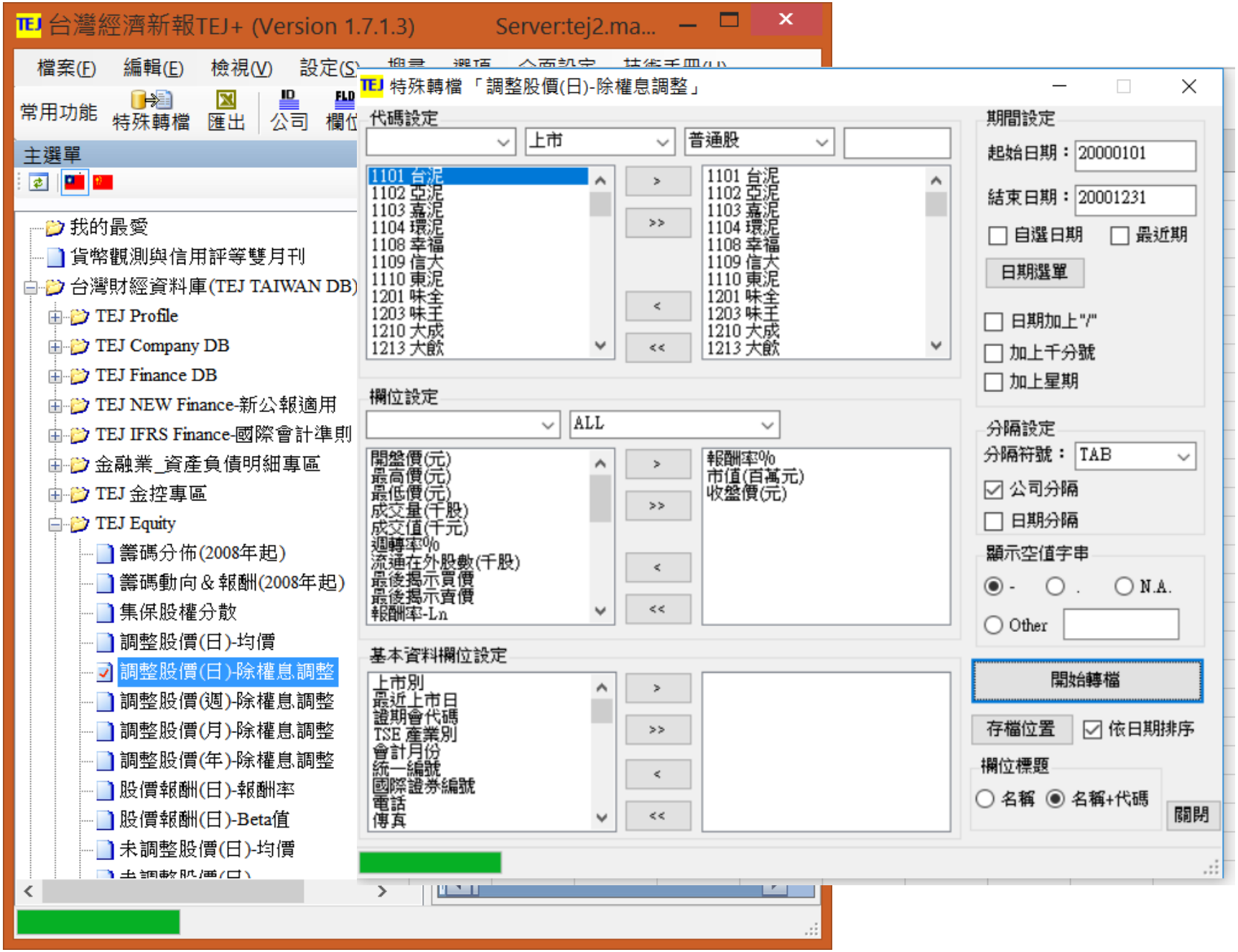
The Market Model

- Assume R_f is a constant = 0.
- We have the following empirical model (Market Model):
- $R_i = \alpha_i + \beta_i R_m + \epsilon_i$
- R_i: Return of stock i
- R_m: Market return
- ϵ_i: Noise

- Meaning of α_i and β_i
- α_i: Stock return when the market return is 0.
- β_i: Security Beta, systematic risk; the sensitivity of a stock to market return.

Data Analysis Steps

- Running the model for every stock-year using daily returns.
- Market return: Need to download market return first (台灣股票加權指數報酬)
- Stock return: Download daily return of all stocks and compute regression model for each stock.
- 資料來源：台灣經濟新報
- 頻率：日資料 (使用除權息調整的資料)
- 包含股票：所有普通股


台灣經濟新報TEJ+ (Version 1.7.1.3)
Server:tej2.ma...
檔案(F)
編輯(E)
檢視(V)
設定(S)
常用功能
特殊轉檔
匯出
公司
主選單
我的最愛
貨幣觀測與信用評等雙月刊
台灣財經資料庫(TEJ TAIWAN DB)
TEJ Profile
TEJ Company DB
TEJ Finance DB
TEJ NEW Finance-新公報適用
TEJ IFRS Finance-國際會計準則
金融業_資產負債明細專區
TEJ 金控專區
TEJ Equity
籌碼分佈(2008年起)
籌碼動向 & 報酬(2008年起)
集保股權分散
調整股價(日)-均價
調整股價(日)-除權息調整
調整股價(週)-除權息調整
調整股價(月)-除權息調整
調整股價(年)-除權息調整
股價報酬(日)-報酬率
股價報酬(日)-Beta值
未調整股價(日)-均價
特殊轉檔「調整股價(日)-除權息調整」
代碼設定
上市
普通股
1101 台泥
1102 亞泥
1103 嘉泥
1104 環泥
1108 幸福
1109 信大
1110 東泥
1201 味全
1203 味王
1210 大成
1213 大飲
欄位設定
ALL
開盤價(元)
最高價(元)
最低價(元)
成交量(千股)
成交值(千元)
週轉率%
流通在外股數(千股)
最後揭示買價
最後揭示賣價
報酬率-Ln
報酬率%
市值(百萬元)
收盤價(元)
基本資料欄位設定
上市別
最近上市日
證期會代碼
TSE 產業別
會計月份
統一編號
國際證券編號
電話
傳真
期間設定
起始日期：20000101
結束日期：20001231
自選日期
最近期
日期選單
日期加上"/"
加上千分號
加上星期
分隔設定
分隔符號：TAB
公司分隔
日期分隔
顯示空值字串
N.A.
Other
開始轉檔
存檔位置
依日期排序
欄位標題
名稱
名稱+代碼
關閉

我想看看我下載的資料

- 使用Notepad++…

證券代碼	簡稱	年月日	報酬率%	市值(百萬元)	收盤價(元)
COID	Name	MDATE	ROI	MV	CLOSE
1101	台泥	20160104	-4.2125	96550	24.14
1102	亞泥	20160104	-4.1971	88237	25.29
1103	嘉泥	20160104	-1.2605	7282	9.29
1104	環泥	20160104	-3.7778	13602	18.90
1108	幸福	20160104	-0.9901	4047	8.61
1109	信大	20160104	-0.4975	3789	9.71
1110	東泥	20160104	-3.7975	8694	14.90
1201	味全	20160104	-1.4286	8729	17.25
1203	味王	20160104	0.0000	5496	21.88
1210	大成	20160104	0.2415	15280	20.23
1213	大飲	20160104	-0.3300	777	13.72
1215	卜蜂	20160104	0.4348	6190	21.10
1216	統一	20160104	-1.8215	306260	50.40
1217	愛之味	20160104	-5.9020	4116	8.32
1218	泰山	20160104	-2.0747	4169	11.80
1219	福壽	20160104	1.0239	4718	14.17
1220	台榮	20160104	-0.9302	1885	9.59

Issues

- The data is “TAB” separated, not “Comma” separated.
- Two head lines, one Chinese, one English.
- Data order is not suitable for our analysis:
  - Current order is by date, then by stock
  - A better way is to order by stock, then by date.
- Still need to have market return data.
  - Need to “merge” market return with stock return by date.

Data Processing Steps

1. Preprocess: Remove Chinese headline, convert to standard CSV file, remove all extra spaces.
2. Sort data by stock and then by date.
3. Prepare market return data
4. For each stock:
    1. Merge stock return with market data by date.
    2. Run regression.
    3. Record the result.

Preprocessing (Step 1)

- We need to read and write file.
- We need to read and write CSV files.

- The process of *opening* a file involves associating a file on disk with a variable.
- We can manipulate the file by manipulating this variable.
  - Read from the file
  - Write to the file ☼

File Processing

- When done with the file, it needs to be *closed*. Closing the file causes any outstanding operations and other bookkeeping for the file to be completed.
- In some cases, not properly closing a file could result in data loss.
- Typical file manipulation routine:
  - File opened
  - Read or write contents from/to the file
  - Close the file ☼

File Processing

- Working with files in Python
  - Associate a file with a variable using the open function <filevar> = open(<name>, <mode>, `encoding = <encoding>`)
  - Name is a string with the actual file name on the disk.
  - <filevar> is often called “file handler”
  - For text file, the mode is either ‘r’ or ‘w’ depending on whether we are reading or writing the file.
  - For non-text files, the mode is “rb” or “wb” for reading or wrting the file
  - <encoding> is the encoding to be used, default to system setting.
  - Example: infile = open("numbers.dat", "r")

File Processing

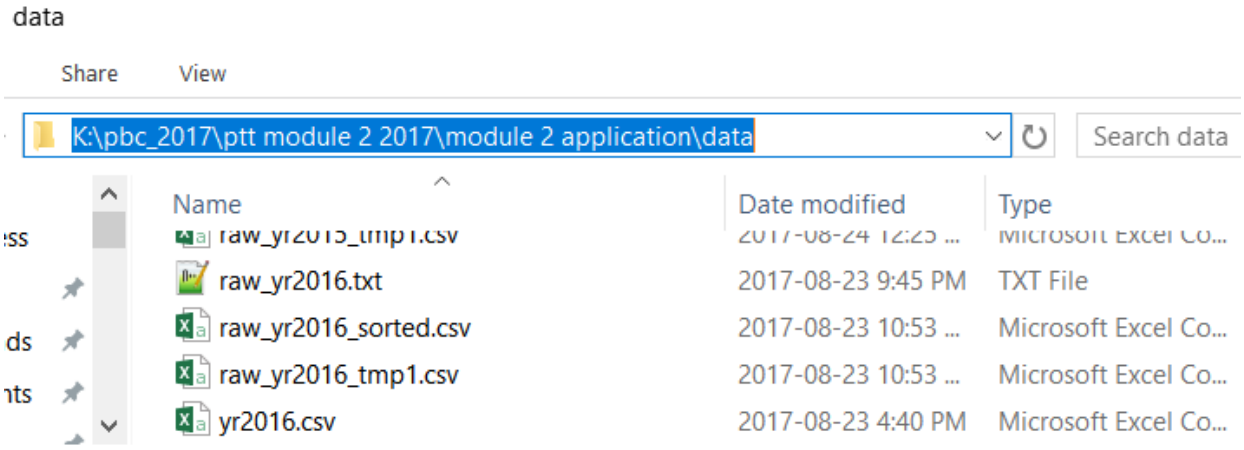
- Let's try this out.

```
stockfn = "raw_yr2016.txt"
fh1 = open(stockfn, 'r')
Traceback (most recent call last):
  File "<input>", line 1, in <module>
FileNotFoundError: [Errno 2] No such file or
directory: 'raw_yr2016.txt'
```

- Failed! Why?
- Python cannot find the file?

File Name and Path

- You need to specify the full path (absolute path; 絕對路徑) so that Python can always access the file correctly.
- Absolute path can be found by opening the folder containing the file, and clicking the folder name.
- In this example, the absolute path is:
- K:\pbc_2017\ptt module 2 2017\module 2 application\data\raw_yr2016.txt



File Name and Path (Cont'd.)

- For Windows Users: Because of historical reason, Windows System use backslash (\) in file path. Other operating systems use slack (/).
- Backslash has a special meaning in string representation.
- Backslash is "escape character."
- The character following escape character are interpreted differently.
- For example, if you are using double quote, and you need to define a string with double quote, then you can use backslash to achieve this.

```
>>> str1 = "A \"test\" string"
>>> print(str1)
A "test" string
```

Escaping Escape Character

- In Python, you cannot use Windows absolute path directly. Instead, you need to change backslash to double backslash.
- E.g. K:\pbc_2017\ptt module 2 2017\module 2 application\data\raw_yr2016.txt
- ➔ K:\\pbc_2017\\ptt module 2 2017\\module 2 application\\data\\raw_yr2016.txt

```
>>> fn0 = "K:\pbc_2017\ptt module 2 2017\module 2 application\data\raw_yr2016.txt"
>>> print(fn0)
K:\pbc_2017\ptt module 2 2017\module 2 application\data
aw_yr2016.txt
>>> fn1 = "K:\\pbc_2017\\ptt module 2 2017\\module 2 application\\data\\raw_yr2016.txt"
>>> print(fn1)
K:\pbc_2017\ptt module 2 2017\module 2 application\data\raw_yr2016.txt
```

Opening and Reading Files

- Now we can read the file!
- cp950 is big5!

```
>>> stockfn = "K:\\pbc_2017\\ptt module 2 2017\\module 2 application\\data\\raw_yr2016.txt"
>>> fh1 = open(stockfn, 'r', encoding = 'cp950')
>>> aline=fh1.readline()
>>> print(aline)
證券代碼  簡稱 年月日     報酬率%  市值(百萬元)  收盤價(元)

>>> aline=fh1.readline()
>>> print(aline)
COID Name MDATE    ROI  MV   CLOSE

>>> fh1.close()
```

Reading CSV Files

- We want to interpret the file as a CSV file.
- But there are a few differences:
- 1. We want to use the second line as the heading.
- 2. We need to interpret TAB as the delimitate character (分隔字元).
- Python has built-in CSV processing library (import csv).
- Create a csv reader object by passing file handler to csv.reader.
- E.g.: `reader2 = csv.reader(fh1, delimiter='\t')`

```
>>> import csv
>>> stockfn = "K:\\pbc_2017\\ptt module 2 2017\\module 2 application\\data\\raw_yr2016.txt"
>>> #set newline='' for csv processing
>>> fh1 = open(stockfn, 'r', encoding = 'cp950', newline='')
>>> cheader=fh1.readline()
>>> reader2 = csv.reader(fh1, delimiter='\t')
>>> print(next(reader2))
['COID', 'Name', 'MDATE', 'ROI', 'MV', 'CLOSE']
>>> print(next(reader2))
['1101   ', '台泥          ', '20160104', '  -4.2125', ' 96550', '   24.14']
>>> print(next(reader2))
['1102   ', '亞泥          ', '20160104', '  -4.1971', ' 88237', '   25.29']
>>> fh1.close()
```

Removing Extra Space, and Save

- Recall that `next(reader2)` returns a list of strings.
- We want to strip all extra white spaces, and save the result to a different file.
- To do so, we need to first create a output file:
- ```
  fh3 = open(stockfn_tmp1, 'w', encoding = 'utf-8', newline='')
  writer3 = csv.writer(fh3)
  ```
- For each row read from the original file, remove extra space for each element in the list:
- ```
  arow = map(lambda x: x.strip(), arow)
  writer3.writerow(arow)
  ```

```
import csv
stockfn = "K:\\pbc_2017\\ptt module 2 2017\\module 2 application\\data\\raw_yr2016.txt"
#set newline='' for csv processing
fh1 = open(stockfn, 'r', encoding = 'cp950', newline='')
cheader=fh1.readline()
reader1 = csv.reader(fh1, delimiter='\t')
#create output file
stockfn_tmp1 = "K:\\pbc_2017\\ptt module 2 2017\\module 2 application\\data\\raw_yr2016_tmp1.csv"
fh3 = open(stockfn_tmp1, 'w', encoding = 'utf-8', newline='')
writer3 = csv.writer(fh3)
for arow in reader1:
    arow = map(lambda x: x.strip(), arow)
    writer3.writerow(arow)
fh3.close()
fh1.close()
```

Output File

- First step completed.

```
COID,Name,MDATE,ROI,MV,CLOSE
1101,台泥,20160104,-4.2125,96550,24.14
1102,亞泥,20160104,-4.1971,88237,25.29
1103,嘉泥,20160104,-1.2605,7282,9.29
1104,環泥,20160104,-3.7778,13602,18.90
1108,幸福,20160104,-0.9901,4047,8.61
1109,信大,20160104,-0.4975,3789,9.71
1110,東泥,20160104,-3.7975,8694,14.90
1201,味全,20160104,-1.4286,8729,17.25
1203,味王,20160104,0.0000,5496,21.88
1210,大成,20160104,0.2415,15280,20.23
```

Sorting CSV File

- We are going to use an external library (csvsorter) to do the job.
- Run: pip3 install csvsorter
- We will be able to import csvsorter and use its functions in our program.

```
C:\Users\hsinminlu>pip3 install csvsorter
Collecting csvsorter
  Downloading csvsorter-1.4.tar.gz
Installing collected packages: csvsorter
  Running setup.py install for csvsorter ... done
Successfully installed csvsorter-1.4
```

More About CSVSORTER

- CSVSORTER partitions a large CSV file, sort them by pieces, and combine.
- It will write temporary files at your working directory.
- To see your working directory,

```
>>> import os
>>> os.getcwd()
'C:\\Program Files (x86)\\Notepad++‘
```

- If running from Notepad++, then the working directory is where you installed Notepad++.
- You will not have write permission in this folder.

Working Directory

- We need to set the working directory to somewhere else.
- For example, the directory of your project.

```
>>> wd="K:\\pbc_2017\\ptt module 2 2017\\module 2 application"
>>> os.chdir(wd)
>>> cwd = os.getcwd()
>>> print("Current working directory:", cwd)
Current working directory: K:\pbc_2017\ptt module 2 2017\module 2 application
```

CSVSORTER

```
1 COID,Name,MDATE,ROI,MV,CLOSE
2 1101,台泥,20160104,-4.2125,96550,24.14
3 1102,亞泥,20160104,-4.1971,88237,25.29
4 1103,嘉泥,20160104,-1.2605,7282,9.29
```

- Specify which columns are used to sort
- `>>> import csvsorter`
- `>>> stockfn_tmp1 = "K:\\pbc_2017\\ptt module 2 2017\\module 2 application\\data\\raw_yr2016_tmp1.csv"`
- `>>> stockfn_sorted = "K:\\pbc_2017\\ptt module 2 2017\\module 2 application\\data\\raw_yr2016_sorted.csv"`
- `>>> csvsorter.csvsort(stockfn_tmp1, [0,2], output_filename=stockfn_sorted, has_header=True)`
- `Merging 1 splits`

Sorted CSV File

- Step 2 completed.

```
COID,Name,MDATE,ROI,MV,CLOSE
1101,台泥,20160104,-4.2125,96550,24.14
1101,台泥,20160105,0.9560,97473,24.37
1101,台泥,20160106,-1.3258,96181,24.05
1101,台泥,20160107,4.9904,100980,25.25
1101,台泥,20160108,-1.2797,99688,24.92
1101,台泥,20160111,-3.7037,95996,24.00
1101,台泥,20160112,0.3846,96365,24.09
1101,台泥,20160113,2.4904,98765,24.69
1101,台泥,20160114,0.0000,98765,24.69
1101,台泥,20160115,0.5607,99319,24.83
1101,台泥,20160118,-1.1152,98211,24.55
```

Prepare Market Data

- Download market return data, convert to CSV by Excel.
- Keep MDATE (date) and MKT (daily market return).
- How are we going to use MKT?
- We need to “merge” with stock return by date.
- How can we do this efficiently?
- We can use the dictionary structure.
- ➔ Use MDATE as key and MKT as value.
- ➔ Easily find MKT of matching date.

	A	B	C
	COID	MDATE	MKT
9	Y9999	20160531	-0.0033
0	Y9999	20160601	0.7213
1	Y9999	20160602	-0.4785
2	Y9999	20160603	0.3663
3	Y9999	20160604	0.049
4	Y9999	20160606	0.0645
5	Y9999	20160607	0.963
6	Y9999	20160608	0.4099
7	Y9999	20160613	-2.0568
8	Y9999	20160614	0.4674
9	Y9999	20160615	0.3527
0	Y9999	20160616	-1.304
1	Y9999	20160617	0.8705
2	Y9999	20160620	0.6751

Market Return to Dictionary

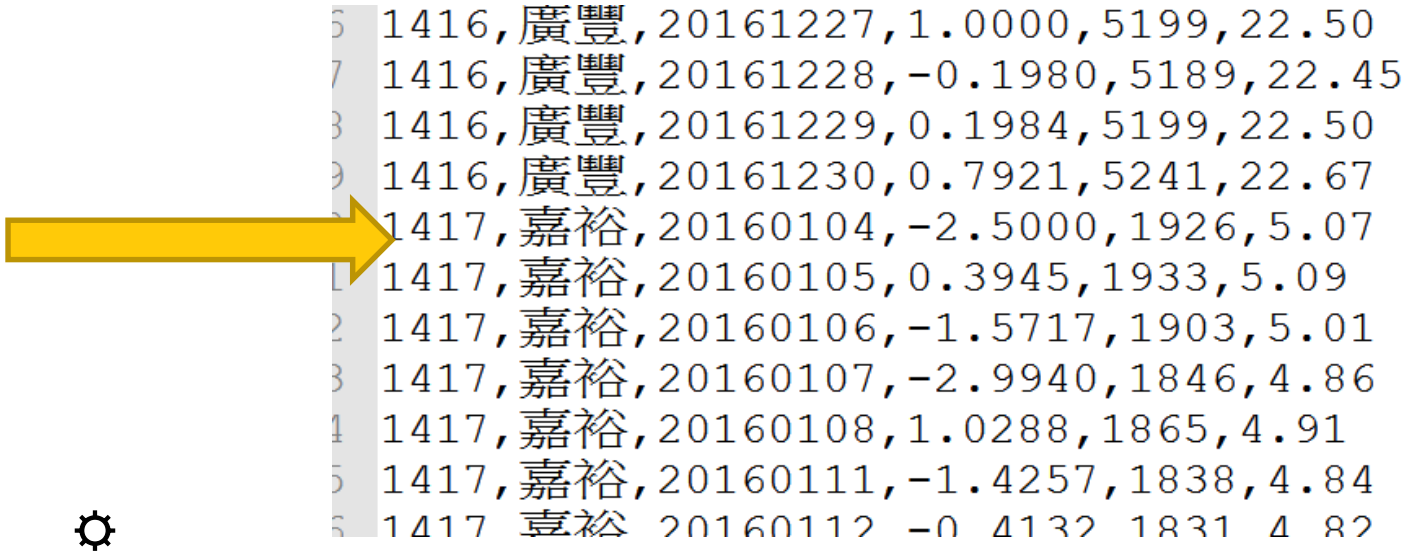
- csv.DictReader(): allow access columns by its names.

```
import csv
mktfn = "K:\\pbc_2017\\ptt module 2 2017\\module 2
application\\data\\mkt1996_2016.csv"

fh1 = open(mktfn, 'r', newline='')
reader1 = csv.DictReader(fh1)
mktret = dict()
for arow in reader1:
    mktret[arow['MDATE']] = float(arow['MKT'])
fh1.close()
print("Read %d market return data" % len(mktret))
```

- Output: Read 5345 market return data
- Get market return by:
- >>> mktret['20160105']
- -0.4825

Running Regression

- We are going to read in stock return data line-by-line.
- Since the data is sorted by stock ID, and then by date, we know that when the stock ID is different from the previous line, then we encountered a new stock.
- How do we know that current stock has finished?
  - Need to remember the stock ID of the previous line!



Loop Through the Sorted CSV File

- Again, create a CSV reader.
- This time, use csv.DictReader()
  - Allow access elements by column name.
- ```
  fh4 = open(stockfn_sorted, 'r', encoding = 'utf-8', newline='')
  reader2 = csv.DictReader(fh4)
  # loop through files…
  for arow in reader2:
      # process return
  fh4.close()
  ```
  ```

Prepare Variables to Store Stock Returns and Dates

- We are going to use lists to store stock returns (sret) and dates (sdate) for each stock.
- Simply append to the end of the list.
- Need to detect the end of one stock data (using last_coid).
-
```
#to store stock returns and dates
sret=[]
sdate=[]
last_coid = ""
for arow in reader2:
    this_coid = arow["COID"].strip()
    this_name = arow['Name'].strip()
    if(this_coid) != last_coid:
        if(len(sret) > minlen):
            #run regression here
        #reset sret and sdate
        sret = [float(arow['ROI'].strip())]
        sdate = [arow['MDATE'].strip()]
    else:
        sret.append(float(arow['ROI'].strip()))
        sdate.append(arow['MDATE'].strip())
    last_coid = this_coid
    last_name = this_name
fh4.close()
```

Prepare to Run Regression

- If we encountered a different stock ID, then its time to run regression for the previous stock ID.
- Pass sret, sdate, and mktret to compute_model().
- Need to reset sret and sdate so that the next stock has a new start!

```
if(this_coid) != last_coid:
    if(len(sret) > minlen):
        print("Run regression for COID:", last_coid)
        out1 = compute_model(sret, sdate, mktret)
        #record out1 here
    #reset sret and sdate
    sret = [float(arow['ROI'].strip())]
    sdate = [arow['MDATE'].strip()]
else:
    sret.append(float(arow['ROI'].strip()))
    sdate.append(arow['MDATE'].strip())
```

Creating the market return list

- Using date to extract market return

```
>>> sdate = ['20160118', '20160119', '20160120',
'20160121', '20160122']
... xlist = []
... for i in range(0, len(sdate)):
...     xlist.append(mktret[sdate[i]])
...
>>> xlist
[0.6335, 0.5595, -1.983, -0.456, 1.2026]
>>> mktret['20160119']
0.5595
```

compute_model()

- We need to construct the market return list first before running regression.
- ```
  def compute_model(ylist, sdate, mktret):
      """ylist: list of stock return
         sdate: list of return dates
         mktret: market return dictonary"""
      xlist = []
      for i in range(0, len(sdate)):
          xlist.append(mktret[sdate[i]])
      if len(xlist) != len(ylist):
          raise Exception("Data lenght Error!")
      return simple_reg(xlist, ylist)
  ```
- `raise Exception()` cause Python to report Error!


  ```

simple_reg()

- New we are ready to run regression.
- Model: $y_{\mathrm{i}} = \alpha + \beta x_i + \epsilon_i$
- $\hat{\beta} = \frac{\sum_{i=1}^{N}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{N}(x_i - \bar{x})}$
- $\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$
- $e_i = y_i - \hat{\alpha} - \hat{\beta}x_i$
- $s = \sqrt{\frac{\sum_{i=1}^{N} e_i^2}{N-2}}$
- $R^2 = 1 - \frac{\sum_{i=1}^{N} e_i^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$

simple_reg()

- Parameters: xlist and ylist
- `def simple_reg(xlist, ylist):`
  `#function def here…`
- `return [alpha, beta, s, r2]`
- Return a list of α, β, s, R^2

```
>>> xlist = [-0.4825, -1.0491, -1.7312,  0.5337, -1.3371, -0.2564, 0.7229, -1.0445,0.2471]
>>> ylist = [0.9560, -1.3258,  4.9904, -1.2797, -3.7037, 0.3846, 2.4904, 0.0000, 0.5607 ]
>>> simple_reg(xlist, ylist)
[0.3333697974835929 7, -0.016504474005063087, 2.6334217234108612, 3.363998221928011e-05]
```

Record Result For Each Stock

- Before entering the for loop
- ```
  coidlist = []
  namelist = []
  alphalist = []
  betalist = []
  slist = []
  r2list = []
  ```
- After running regression for each stock:
- ```
  out1 = compute_model(sret, sdate, mktret)
  coidlist.append(last_coid)
  namelist.append(last_name)
  alphalist.append(out1[0])
  betalist.append(out1[1])
  slist.append(out1[2])
  r2list.append(out1[3])
  ```

Remember to Process the Last Stock

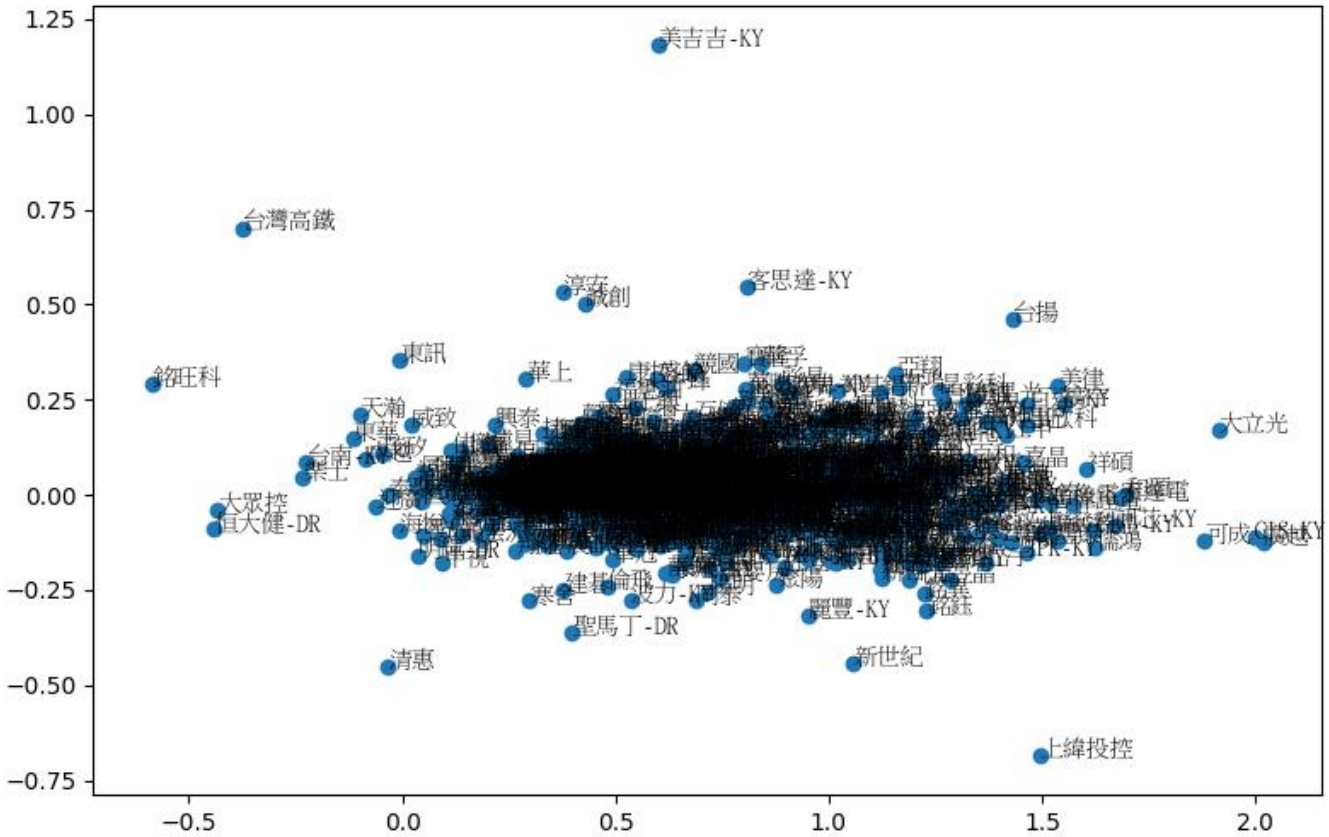
- Still need to run regression for the last stock after finish looping.
- Outside of the for loop:
- ```
  #the last stock
  if(len(sret) > minlen):
      print("Run regression for COID:", last_coid)
      out1=compute_model(sret, sdate, mktret)
      coidlist.append(last_coid)
      namelist.append(last_name)
      alphalist.append(out1[0])
      betalist.append(out1[1])
      slist.append(out1[2])
      r2list.append(out1[3])
  ```


  ```

Putting Everything Together

- Running the program:
- Read 5345 market return data
- Merging 1 splits
- Run regression for COID: 1101
- Run regression for COID: 1102
- Run regression for COID: 1103
- …
- Run regression for COID: 9958
- Run regression for COID: 9958
- **Average R2= 0.132214; nstock=906**

Plotting Result

- Using matplotlib
- ```
  import matplotlib.pyplot as plt
  from matplotlib.font_manager import FontProperties

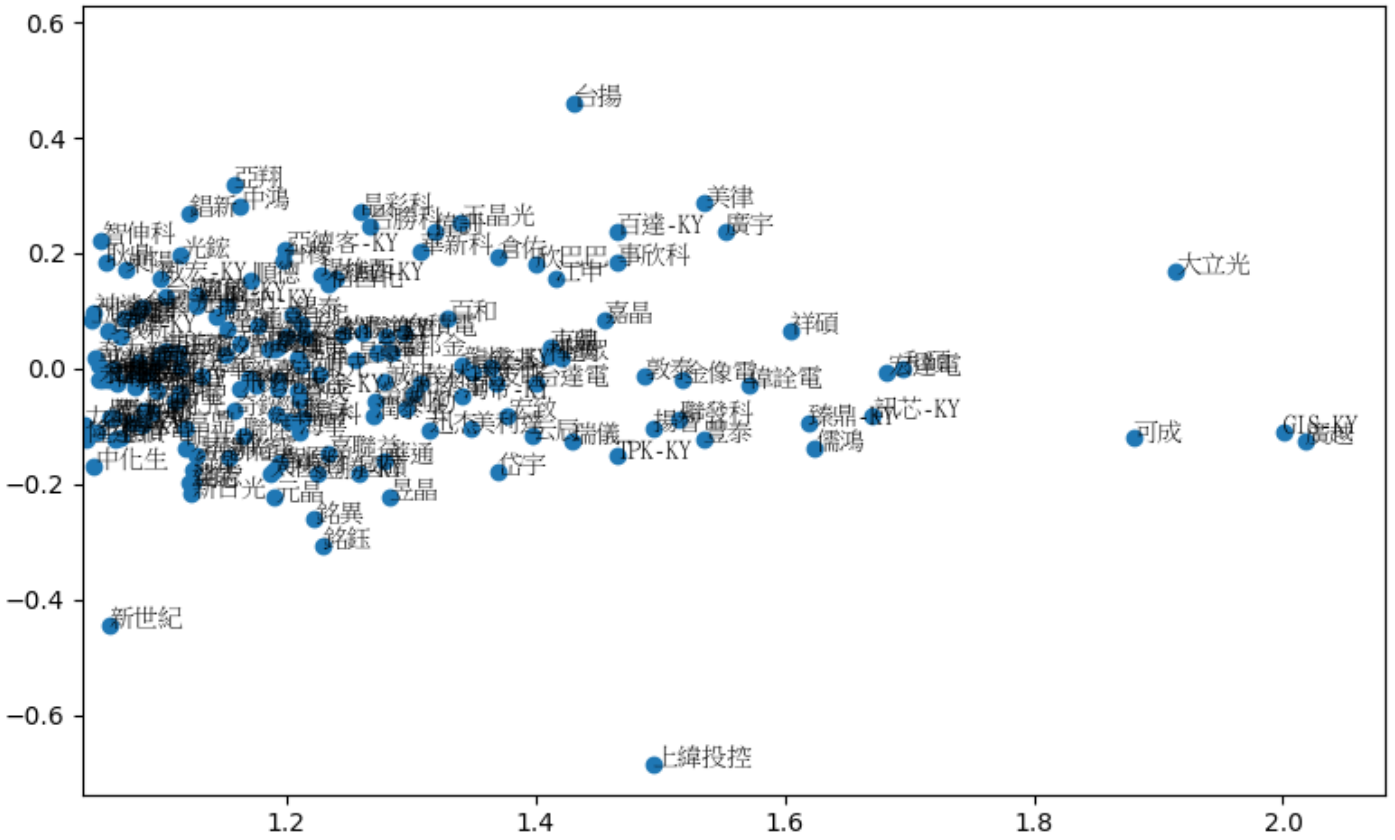
  ChineseFont2 = FontProperties(fname =
  'C:\\Windows\\Fonts\\mingliu.ttc')
  fig, ax = plt.subplots()
  ax.scatter(betalist, alphalist)

  for i, txt in enumerate(namelist):
      ax.annotate(txt, (betalist[i],alphalist[i]),
  fontproperties = ChineseFont2)
  plt.show()
  ```


  ```

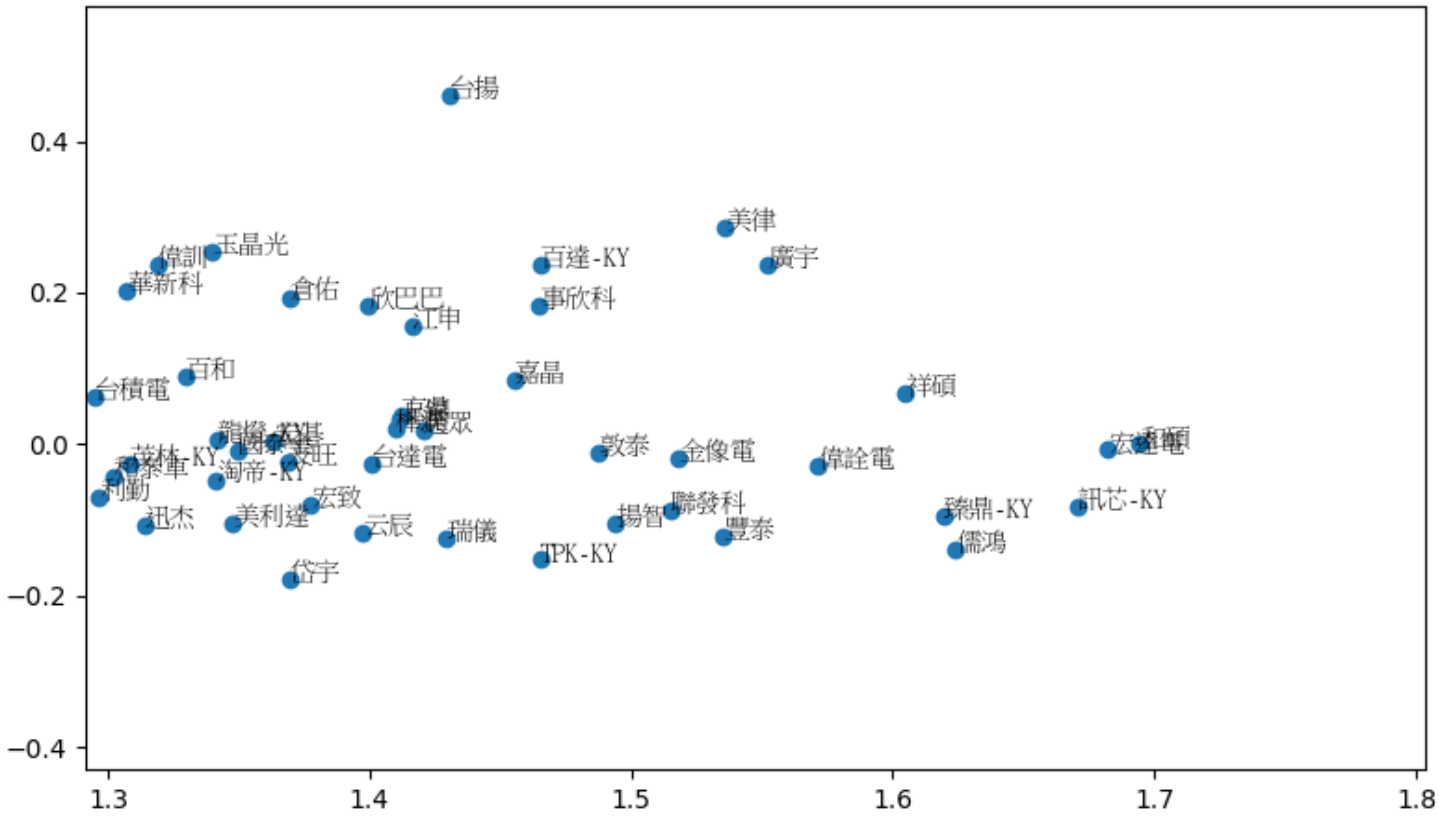
Plotting the Result (x: beta; y: alpha)



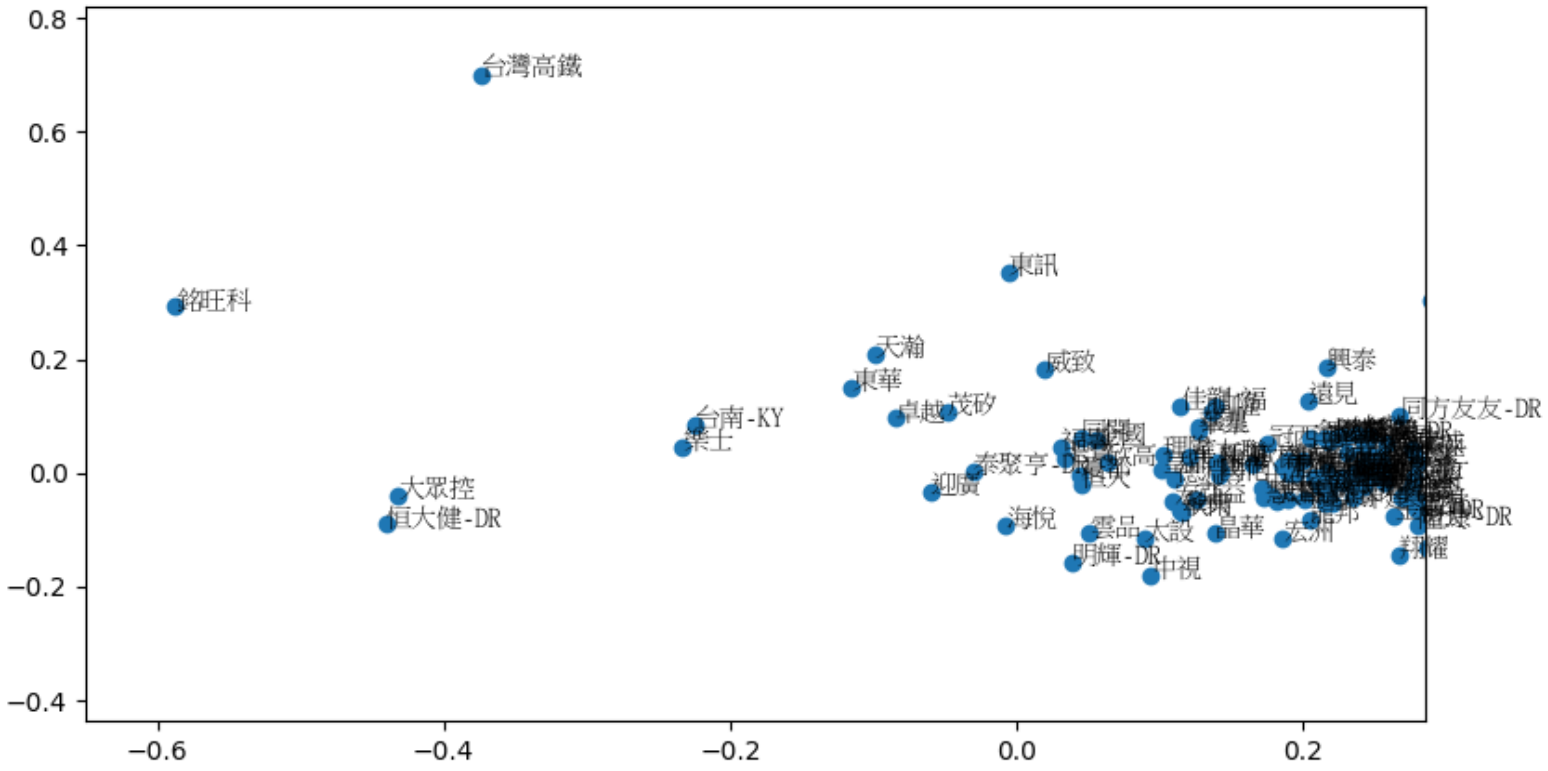
Plotting the Result (x: beta; y: alpha)



Plotting the Result (x: beta; y: alpha)



Plotting the Result (x: beta; y: alpha)

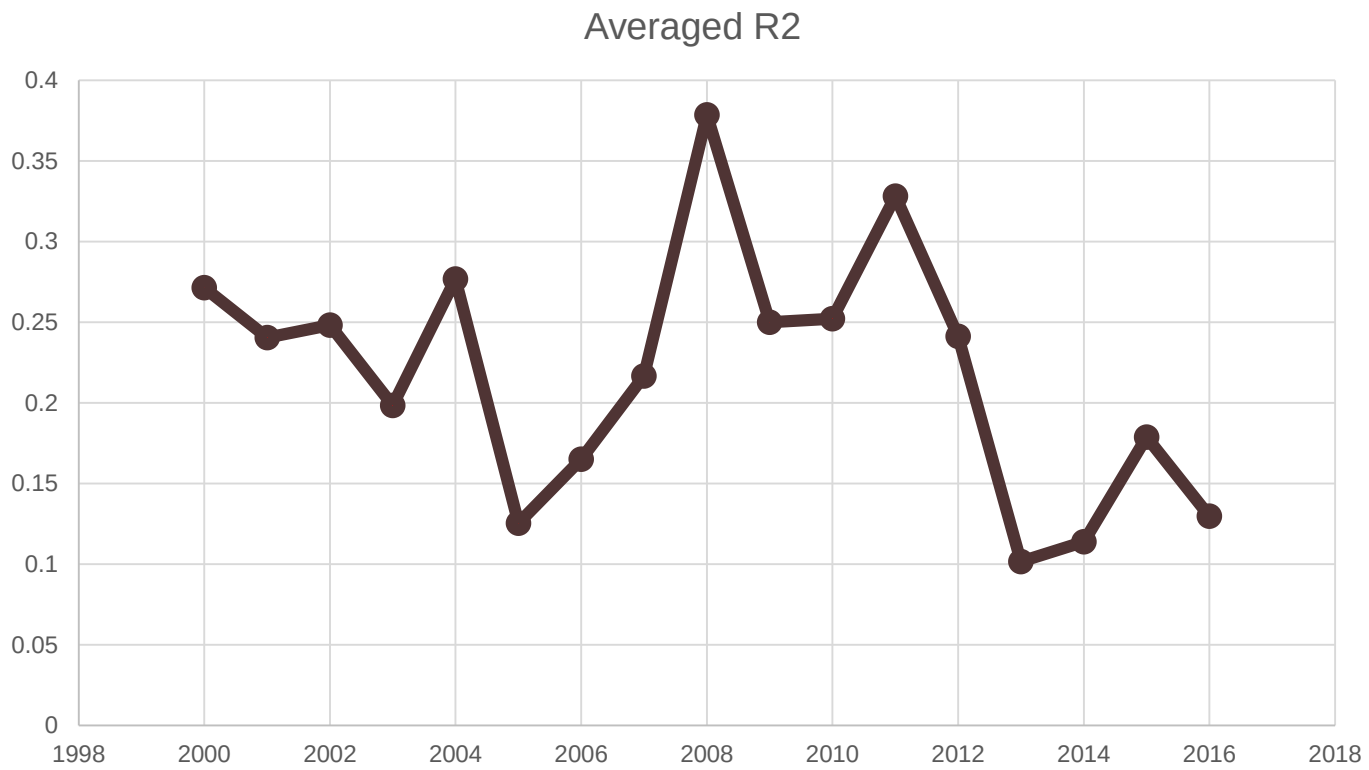


Variance Decomposition

- Based on the market model: $R_i = \alpha_i + \beta_i R_m + \epsilon_i$
- We know that $Var(R_i) = Var(\hat{R}_i) + Var(\epsilon_i)$
- In words: Stock variance can be decomposed into two parts: systematic risk $Var(\hat{R}_i)$ and idiosyncratic risk $Var(\epsilon_i)$.
- Systematic risk is the variation that is linked to the market.
- Idiosyncratic risk is the variation that is not related to the market.
- $R^2 = \frac{Var(\hat{R}_i)}{Var(R_i)}$
- Averaged R^2: Whether stocks move together (driven by market return)
- Higher averaged R^2: stocks are driven by market
- Lower average R^2: stocks are not driven by market

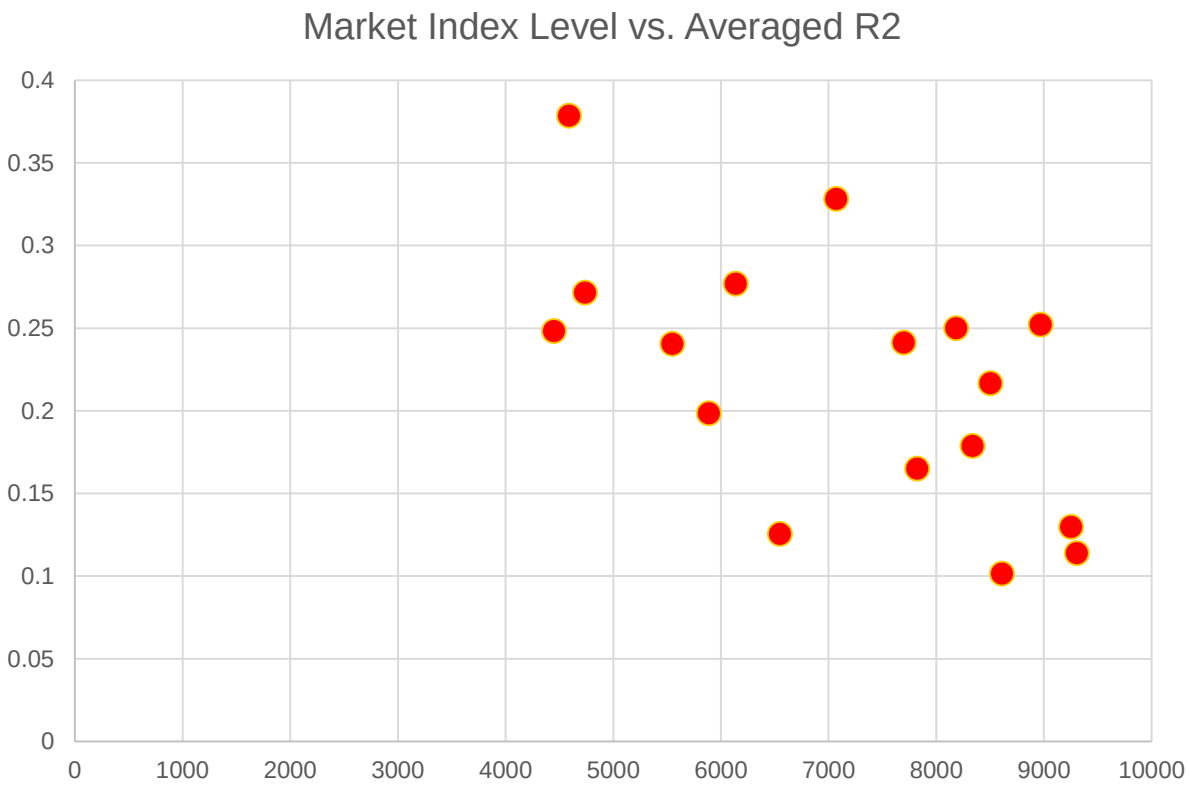
Run the Same Model For Different Years

- Year 2000 to 2016.
- Collect averaged R^2.



Market Index Level vs. Averaged R2

- A clear negative relationship
- Stocks move together when market index is low.



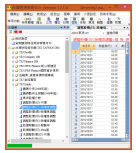
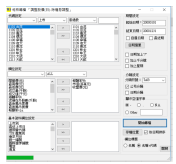
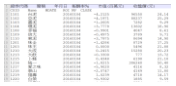
Final Words

- We have demonstrate how to process real world datasets with simple python scripts.
- Because of the pedagogical nature of this course, we have avoid “advanced” libraries and tools that may be more attractive in some situations.
- You will need to explore the landscape before jumping into coding. ☼

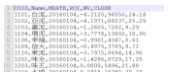
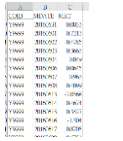
THANK YOU!

Questions?

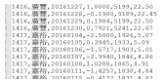
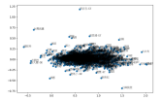
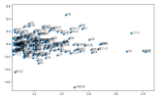
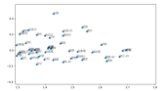
版權聲明

序	頁	作品	版權標章	作者 / 來源
1	5			Taiwan Stock Exchange Corporation ,臺灣資本市場概況-歡迎外資來台投資, p4, 改作: 盧信銘 http://www.twse.com.tw/downloads/zh/investor/foreignInvest/TCMI_CH_1501.pdf 依據著作權法第46、52、65條合理使用 2017/8/30 visited
2	6			Taiwan Stock Exchange Corporation ,臺灣資本市場概況-歡迎外資來台投資, p7, 改作: 盧信銘 http://www.twse.com.tw/downloads/zh/investor/foreignInvest/TCMI_CH_1501.pdf 2017/8/30 visited
3	7			台灣經濟新報, 台灣經濟新報資料庫系統 操作:盧信銘 http://www.tej.com.tw/twsite/Default.aspx?TabId=396 依據著作權法第46、52、65條合理使用 2017/8/30 visited
4	13			台灣經濟新報, 台灣經濟新報資料庫系統 操作:盧信銘 http://www.tej.com.tw/twsite/Default.aspx?TabId=396 依據著作權法第46、52、65條合理使用 2017/8/30 visited
5	13			台灣經濟新報, 台灣經濟新報資料庫系統,操作:盧信銘 http://www.tej.com.tw/twsite/Default.aspx?TabId=396 依據著作權法第46、52、65條合理使用 2017/8/30 visited
6	14			台灣經濟新報, 台灣經濟新報資料庫系統,操作:盧信銘 http://www.tej.com.tw/twsite/Default.aspx?TabId=396 依據著作權法第46、52、65條合理使用 2017/8/30 visited

版權聲明

序	頁	作品	版權標章	作者 / 來源
6	19			Flicker, photographyplayers, IMAG3003(4x4) https://goo.gl/xVwppL 2017/8/24 visited
7	21		CC BY-NC-ND	台灣大學 盧信銘, CC BY-NC-ND 3.0
8	29 33		CC BY-NC-ND	台灣大學 盧信銘, CC BY-NC-ND 3.0
9	30		CC BY-NC-ND	台灣大學 盧信銘, CC BY-NC-ND 3.0
10	34		CC BY-NC-ND	台灣大學 盧信銘, CC BY-NC-ND 3.0
11	35		CC BY-NC-ND	台灣大學 盧信銘, CC BY-NC-ND 3.0

版權聲明

序	頁	作品	版權標章	作者 / 來源
12	37			台灣大學 盧信銘, CC BY-NC-ND 3.0
13	49			台灣大學 盧信銘, CC BY-NC-ND 3.0
14	50			台灣大學 盧信銘, CC BY-NC-ND 3.0
15	51			台灣大學 盧信銘, CC BY-NC-ND 3.0
16	52			台灣大學 盧信銘, CC BY-NC-ND 3.0
17	54			台灣大學 盧信銘, CC BY-NC-ND 3.0

版權聲明

序	頁	作品	版權標章	作者 / 來源
17	55			台灣大學 盧信銘, CC BY-NC-ND 3.0
18	1-61			台灣大學 盧信銘, CC BY-NC-ND 3.0