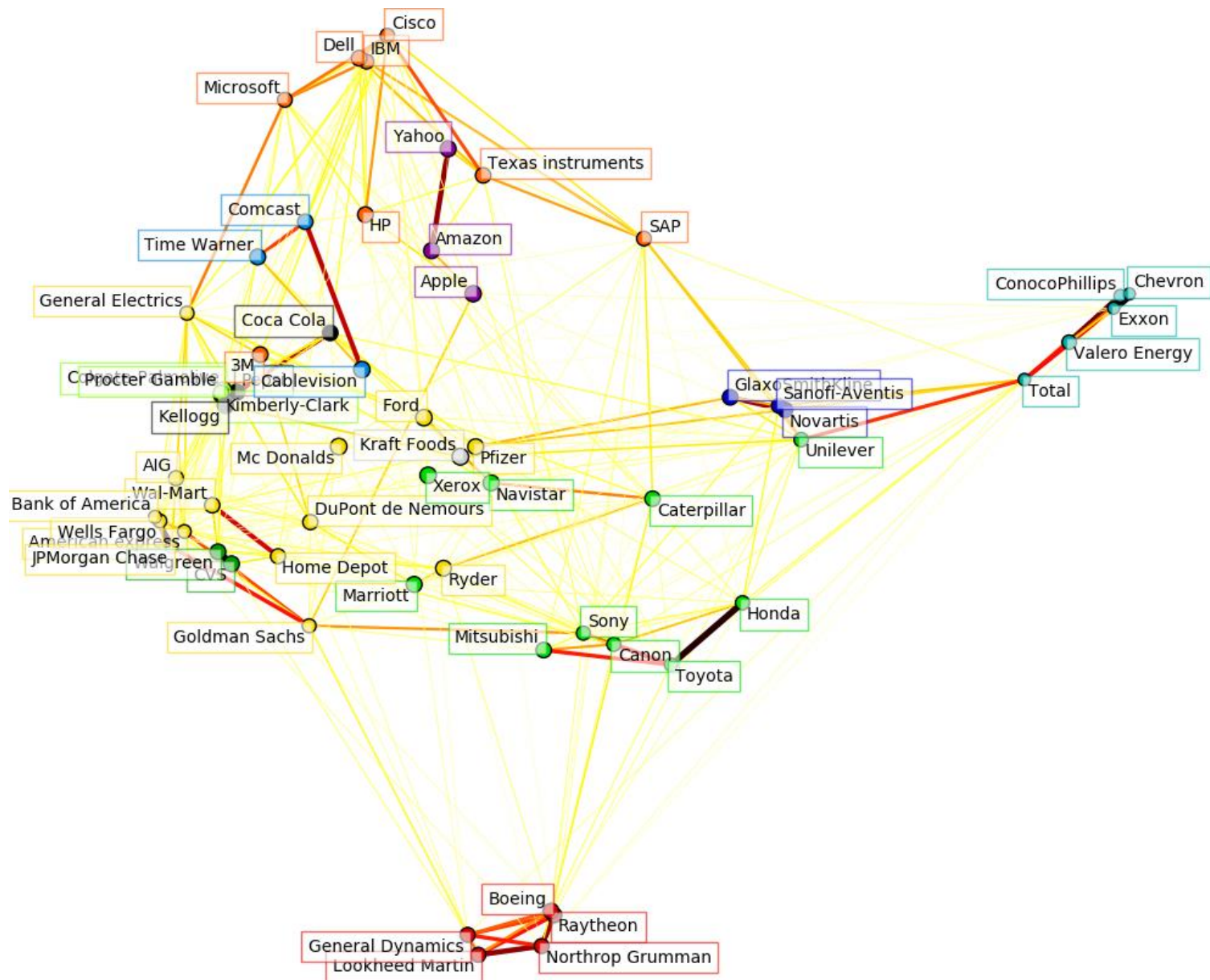
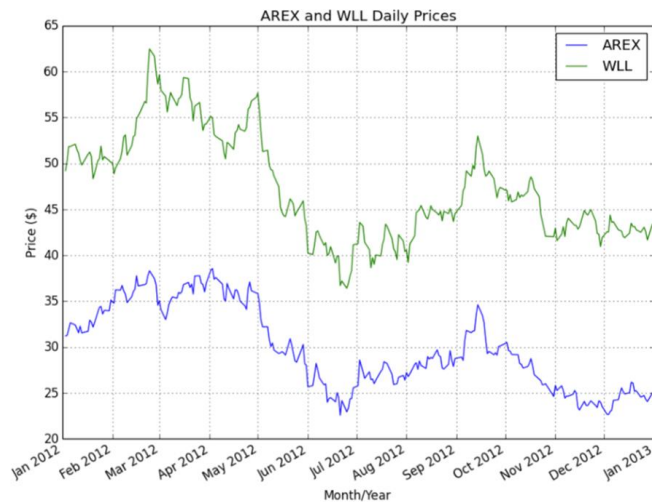


机器学习与量化交易实战

第三课

Outline

- 数据的获得与存储
- 时间序列分析实战



- 从下周开始, 每次课前半小时为上次作业点评并给出解决方案的程序

建立数据库的意义

- 建模
 - 模型建立
 - 模型评估
 - 模型回测
 - 风险控制
- 交易
- 目的：能存，能取

数据的获取

- <http://tushare.org/index.html>
- **Yahoo Finance** - <http://finance.yahoo.com>
- **Google Finance** - <https://www.google.com/finance>
- **QuantQuote** - <https://www.quantquote.com> (S&P500 EOD data only)
- **EODData** - <http://eoddata.com> (requires registration)

存储方式

- CSV
- NoSQL
- SQL
- ○ ○ ○

数据格式

- 交易所信息
- 数据来源
- Ticker/symbol
- 价格
- 企业行为 (stock splits/dividend adjustments)
- 国家假日

容易出错的地方

- 企业行为
- “spikes”：用spike filter
- 缺失数据

搭建你自己的MySQL

- 安装

```
sudo apt-get install mysql-server
```

```
mysql -u root -p
```

```
mysql> CREATE DATABASE securities_master;  
mysql> USE securities_master;
```

```
mysql> CREATE USER 'sec_user'@'localhost' IDENTIFIED BY 'password';  
mysql> GRANT ALL PRIVILEGES ON securities_master.* TO 'sec_user'@'localhost';  
mysql> FLUSH PRIVILEGES;
```

设计股票EOD数据的表

- Exchange
- DataVendor
- Symbol
- DailyPrice

```
CREATE TABLE 'exchange' (  
    'id' int NOT NULL AUTO_INCREMENT,  
    'abbrev' varchar(32) NOT NULL,  
    'name' varchar(255) NOT NULL,  
    'city' varchar(255) NULL,  
    'country' varchar(255) NULL,  
    'currency' varchar(64) NULL,  
    'timezone_offset' time NULL,  
    'created_date' datetime NOT NULL,  
    'last_updated_date' datetime NOT NULL,  
    PRIMARY KEY ('id')  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

```
CREATE TABLE 'data_vendor' (  
  'id' int NOT NULL AUTO_INCREMENT,  
  'name' varchar(64) NOT NULL,  
  'website_url' varchar(255) NULL,  
  'support_email' varchar(255) NULL,  
  'created_date' datetime NOT NULL,  
  'last_updated_date' datetime NOT NULL,  
  PRIMARY KEY ('id')  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

```
CREATE TABLE 'symbol' (  
  'id' int NOT NULL AUTO_INCREMENT,  
  'exchange_id' int NULL,  
  'ticker' varchar(32) NOT NULL,  
  'instrument' varchar(64) NOT NULL,  
  'name' varchar(255) NULL,  
  'sector' varchar(255) NULL,  
  'currency' varchar(32) NULL,  
  'created_date' datetime NOT NULL,  
  'last_updated_date' datetime NOT NULL,  
  PRIMARY KEY ('id'),  
  KEY 'index_exchange_id' ('exchange_id')  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

```
CREATE TABLE 'daily_price' (  
  'id' int NOT NULL AUTO_INCREMENT,  
  'data_vendor_id' int NOT NULL,  
  'symbol_id' int NOT NULL,  
  'price_date' datetime NOT NULL,  
  'created_date' datetime NOT NULL,  
  'last_updated_date' datetime NOT NULL,  
  'open_price' decimal(19,4) NULL,  
  'high_price' decimal(19,4) NULL,  
  'low_price' decimal(19,4) NULL,  
  'close_price' decimal(19,4) NULL,  
  'adj_close_price' decimal(19,4) NULL,  
  'volume' bigint NULL,  
  PRIMARY KEY ('id'),  
  KEY 'index_data_vendor_id' ('data_vendor_id'),  
  KEY 'index_symbol_id' ('symbol_id')  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

python同数据库连接

```
sudo apt-get install libmysqlclient-dev  
pip install mysqlclient
```


- 任务：
 - 从a处获得股票的symbol
 - 从b处获得相应的价格
 - 存在某处。。
-
- 将来可以随时读取

List of S&P 500 companies

From Wikipedia, the free encyclopedia

The **S&P 500 stock market index**, maintained by **S&P Dow Jones Indices**, comprises **505 common stocks** issued by **500 mid an** 75 percent of the American equity market by capitalization. The index is weighted by free-float market capitalization, so more va and the constituent weights are updated and checked on regularly using rules published by S&P Dow Jones Indices. Although tl two share classes of stock from 5 of its component companies.^{[1][2]} The index constituents listed below were current as of July 1:

Contents [hide]
1 S&P 500 Component Stocks
2 Recent and announced changes to the list of S&P 500 Components
3 See also
4 External links
5 References

S&P 500 Component Stocks [edit]

Ticker symbol ✚	Security ✚	SEC filings ✚	GICS Sector ✚	GICS Sub Indust
MMM ✚	3M Company	reports ✚	Industrials	Industrial Conglomerates
ABT ✚	Abbott Laboratories	reports ✚	Health Care	Health Care Equipment & S
ABBV ✚	AbbVie	reports ✚	Health Care	Pharmaceuticals
ACN ✚	Accenture plc	reports ✚	Information Technology	IT Consulting & Other Servi
ATVI ✚	Activision Blizzard	reports ✚	Information Technology	Home Entertainment Softwa
AYI ✚	Acuity Brands Inc	reports ✚	Industrials	Electrical Components & Ec
ADBE ✚	Adobe Systems Inc	reports ✚	Information Technology	Application Software
AAP ✚	Advance Auto Parts	reports ✚	Consumer Discretionary	Automotive Retail
AES ✚	AES Corp	reports ✚	Utilities	Independent Power Produc Traders

- 课件
- Insert_symbols.py

List of S&P 500 companies

From Wikipedia, the free encyclopedia

The **S&P 500 stock market index**, maintained by **S&P Dow Jones Indices**, comprises **505 common stocks** issued by **500 mid an** 75 percent of the American equity market by capitalization. The index is weighted by free-float market capitalization, so more va and the constituent weights are updated and checked on regularly using rules published by S&P Dow Jones Indices. Although tl two share classes of stock from 5 of its component companies.^{[1][2]} The index constituents listed below were current as of July 1:

Contents [hide]

- S&P 500 Component Stocks
- Recent and announced changes to the list of S&P 500 Components
- See also
- External links
- References

S&P 500 Component Stocks [edit]

Ticker symbol ↕	Security ↕	SEC filings ↕	GICS Sector ↕	GICS Sub Indust
MMM ↗	3M Company	reports ↗	Industrials	Industrial Conglomerates
ABT ↗	Abbott Laboratories	reports ↗	Health Care	Health Care Equipment & S
ABBV ↗	AbbVie	reports ↗	Health Care	Pharmaceuticals
ACN ↗	Accenture plc	reports ↗	Information Technology	IT Consulting & Other Servi
ATVI ↗	Activision Blizzard	reports ↗	Information Technology	Home Entertainment Softwa
AYI ↗	Acuity Brands Inc	reports ↗	Industrials	Electrical Components & Ec
ADBE ↗	Adobe Systems Inc	reports ↗	Information Technology	Application Software
AAP ↗	Advance Auto Parts	reports ↗	Consumer Discretionary	Automotive Retail
AES ↗	AES Corp	reports ↗	Utilities	Independent Power Producti Traders

```
def obtain_parse_wiki_snp500():
    """
    Download and parse the Wikipedia list of S&P500
    constituents using requests and BeautifulSoup.
    Returns a list of tuples for to add to MySQL.
    """

    # Stores the current time, for the created_at record
    now = datetime.datetime.utcnow()
    # Use requests and BeautifulSoup to download the
    # list of S&P500 companies and obtain the symbol table
    response = requests.get(
        "http://en.wikipedia.org/wiki/List_of_S%26P_500_companies"
    )
    soup = bs4.BeautifulSoup(response.text)
    # This selects the first table, using CSS Selector syntax
    # and then ignores the header row ([1:])
    symbolslist = soup.select('table')[0].select('tr')[1:]
    # Obtain the symbol information for each
    # row in the S&P500 constituent table
    symbols = []
    for i, symbol in enumerate(symbolslist):
        tds = symbol.select('td')
        symbols.append(
            (
                tds[0].select('a')[0].text, # Ticker
                'stock',
                tds[1].select('a')[0].text, # Name
                tds[3].text, # Sector
                'USD', now, now
            )
        )
    return symbols
```

- 课件
- Insert_symbols.py

```
def insert_snp500_symbols(symbols):  
    """  
    Insert the S&P500 symbols into the MySQL database.  
    """  
  
    # Connect to the MySQL instance  
    db_host = 'localhost'  
    db_user = 'sec_user'  
    db_pass = 'password'  
    db_name = 'securities_master'  
    con = mdb.connect(  
        host=db_host, user=db_user, passwd=db_pass, db=db_name  
    )  
  
    # Create the insert strings  
    column_str = """ticker, instrument, name, sector,  
                    currency, created_date, last_updated_date  
    """  
  
    insert_str = ("%s, " * 7)[: -2]  
    final_str = "INSERT INTO symbol (%s) VALUES (%s)" % \  
        (column_str, insert_str)  
  
    # Using the MySQL connection, carry out  
    # an INSERT INTO for every symbol  
    with con:  
        cur = con.cursor()  
        cur.executemany(final_str, symbols)
```

- 课件
- Price_retrivel.py

```
def get_daily_historic_data_yahoo(
    ticker, start_date=(2000,1,1),
    end_date=datetime.date.today().timetuple()[0:3]
):
    """
    Obtains data from Yahoo Finance returns and a list of tuples.

    ticker: Yahoo Finance ticker symbol, e.g. "GOOG" for Google, Inc.
    start_date: Start date in (YYYY, M, D) format
    end_date: End date in (YYYY, M, D) format
    """
    # Construct the Yahoo URL with the correct integer query parameters
    # for start and end dates. Note that some parameters are zero-based!
    ticker_tup = (
        ticker, start_date[1]-1, start_date[2],
        start_date[0], end_date[1]-1, end_date[2],
        end_date[0]
    )
    yahoo_url = "http://ichart.finance.yahoo.com/table.csv"
    yahoo_url += "?s=%s&a=%s&b=%s&c=%s&d=%s&e=%s&f=%s"
    yahoo_url = yahoo_url % ticker_tup

    # Try connecting to Yahoo Finance and obtaining the data
    # On failure, print an error message.
    try:
        yf_data = requests.get(yahoo_url).text.split("\n")[1:-1]
        prices = []
        for y in yf_data:
            p = y.strip().split(',')
            prices.append(
                (datetime.datetime.strptime(p[0], '%Y-%m-%d'),
                 p[1], p[2], p[3], p[4], p[5], p[6])
            )
    except Exception as e:
        print("Could not download Yahoo data: %s" % e)
    return prices
```

- 课件
- Price_retrivel.py

```
def insert_daily_data_into_db(
    data_vendor_id, symbol_id, daily_data
):
    """
    Takes a list of tuples of daily data and adds it to the
    MySQL database. Appends the vendor ID and symbol ID to the data.

    daily_data: List of tuples of the OHLC data (with
    adj_close and volume)
    """
    # Create the time now
    now = datetime.datetime.utcnow()

    # Amend the data to include the vendor ID and symbol ID
    daily_data = [
        (data_vendor_id, symbol_id, d[0], now, now,
         d[1], d[2], d[3], d[4], d[5], d[6])
        for d in daily_data
    ]

    # Create the insert strings
    column_str = """data_vendor_id, symbol_id, price_date, created_date,
                    last_updated_date, open_price, high_price, low_price,
                    close_price, volume, adj_close_price"""
    insert_str = ("%s, " * 11)[: -2]
    final_str = "INSERT INTO daily_price (%s) VALUES (%s)" % \
        (column_str, insert_str)

    # Using the MySQL connection, carry out an INSERT INTO for every symbol
    with con:
        cur = con.cursor()
        cur.executemany(final_str, daily_data)
```

- 课件
- Retrieving_Data.py

	adj_close_price
price_date	
2015-06-09	526.69
2015-06-10	536.69
2015-06-11	534.61
2015-06-12	532.33
2015-06-15	527.20

```
from __future__ import print_function

import pandas as pd
import MySQLdb as mdb

if __name__ == "__main__":
    # Connect to the MySQL instance
    db_host = 'localhost'
    db_user = 'sec_user'
    db_pass = 'password'
    db_name = 'securities_master'
    con = mdb.connect(db_host, db_user, db_pass, db_name)

    # Select all of the historic Google adjusted close data
    sql = """SELECT dp.price_date, dp.adj_close_price
              FROM symbol AS sym
              INNER JOIN daily_price AS dp
              ON dp.symbol_id = sym.id
              WHERE sym.ticker = 'GOOG'
              ORDER BY dp.price_date ASC;"""

    # Create a pandas dataframe from the SQL query
    goog = pd.read_sql_query(sql, con=con, index_col='price_date')

    # Output the dataframe tail
    print(goog.tail())
```

More on Yahoo finance

```
from __future__ import print_function

import datetime
import pandas.io.data as web

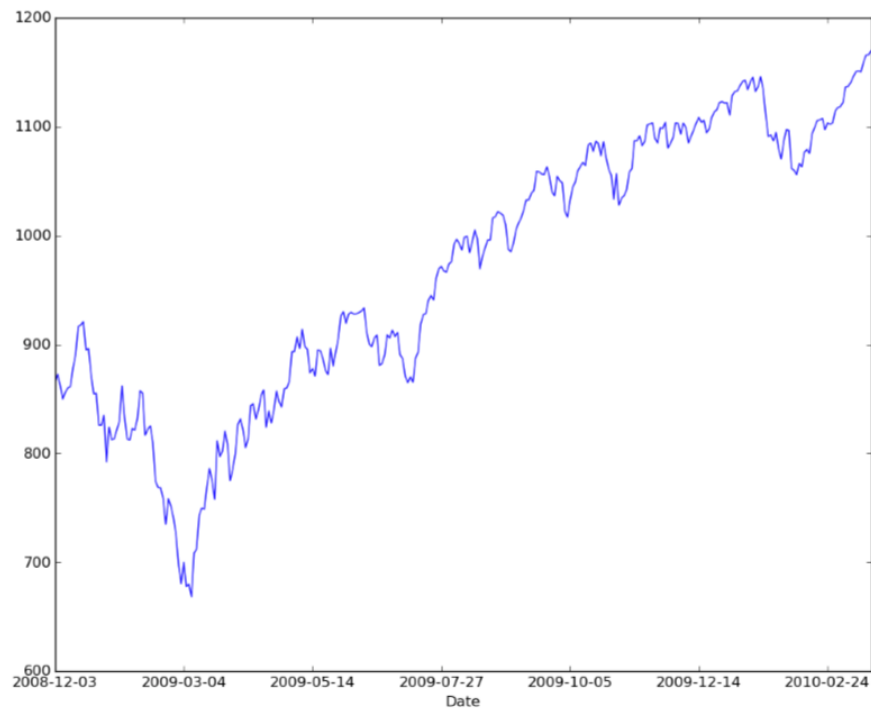
if __name__ == "__main__":
    spy = web.DataReader(
        "SPY", "yahoo",
        datetime.datetime(2007,1,1),
        datetime.datetime(2015,6,15)
    )
    print(spy.tail())
```

Date	Open	High	Low	Close	Volume \
2015-06-09	208.449997	209.100006	207.690002	208.449997	98148200
2015-06-10	209.369995	211.410004	209.300003	210.960007	129936200
2015-06-11	211.479996	212.089996	211.199997	211.649994	72672100
2015-06-12	210.639999	211.479996	209.679993	209.929993	127811900
2015-06-15	208.639999	209.449997	207.789993	209.100006	121425800
	Adj Close				
2015-06-09	208.449997				
2015-06-10	210.960007				
2015-06-11	211.649994				
2015-06-12	209.929993				
2015-06-15	209.100006				

Quandl

- Quandl_data.py

```
def download_contract_from_quandl(contract, dl_dir):  
    """  
    Download an individual futures contract from Quandl and then  
    store it to disk in the 'dl_dir' directory. An auth_token is  
    required, which is obtained from the Quandl upon sign-up.  
    """  
  
    # Construct the API call from the contract and auth_token  
    api_call = "http://www.quandl.com/api/v1/datasets/"  
    api_call += "OFDP/FUTURE_%s.csv" % contract  
    # If you wish to add an auth token for more downloads, simply  
    # comment the following line and replace MY_AUTH_TOKEN with  
    # your auth token in the line below  
    params = "?sort_order=asc"  
    #params = "?auth_token=MY_AUTH_TOKEN&sort_order=asc"  
    full_url = "%s%s" % (api_call, params)  
  
    # Download the data from Quandl  
    data = requests.get(full_url).text  
  
    # Store the data to disk  
    fc = open('%s/%s.csv' % (dl_dir, contract), 'w')  
    fc.write(data)  
    fc.close()
```



```
if __name__ == "__main__":  
    symbol = 'ES'  
  
    # Make sure you've created this  
    # relative directory beforehand  
    dl_dir = 'quandl/futures/ES'  
  
    # Create the start and end years  
    start_year = 2010  
    end_year = 2014  
  
    # Download the contracts into the directory  
    download_historical_contracts(  
        symbol, dl_dir, start_year, end_year  
    )  
  
    # Open up a single contract via read_csv  
    # and plot the settle price  
    es = pd.io.parsers.read_csv(  
        "%s/ESH2010.csv" % dl_dir, index_col="Date"  
    )  
    es["Settle"].plot()  
    plt.show()
```

作业1

利用<http://tushare.org/index.html> 中介绍的api, 爬取<http://tushare.org/trading.html> 页面中的所有交易数据存入本地数据库（或csv文件夹）。

hint: 参考 <http://tushare.org/storing.html>

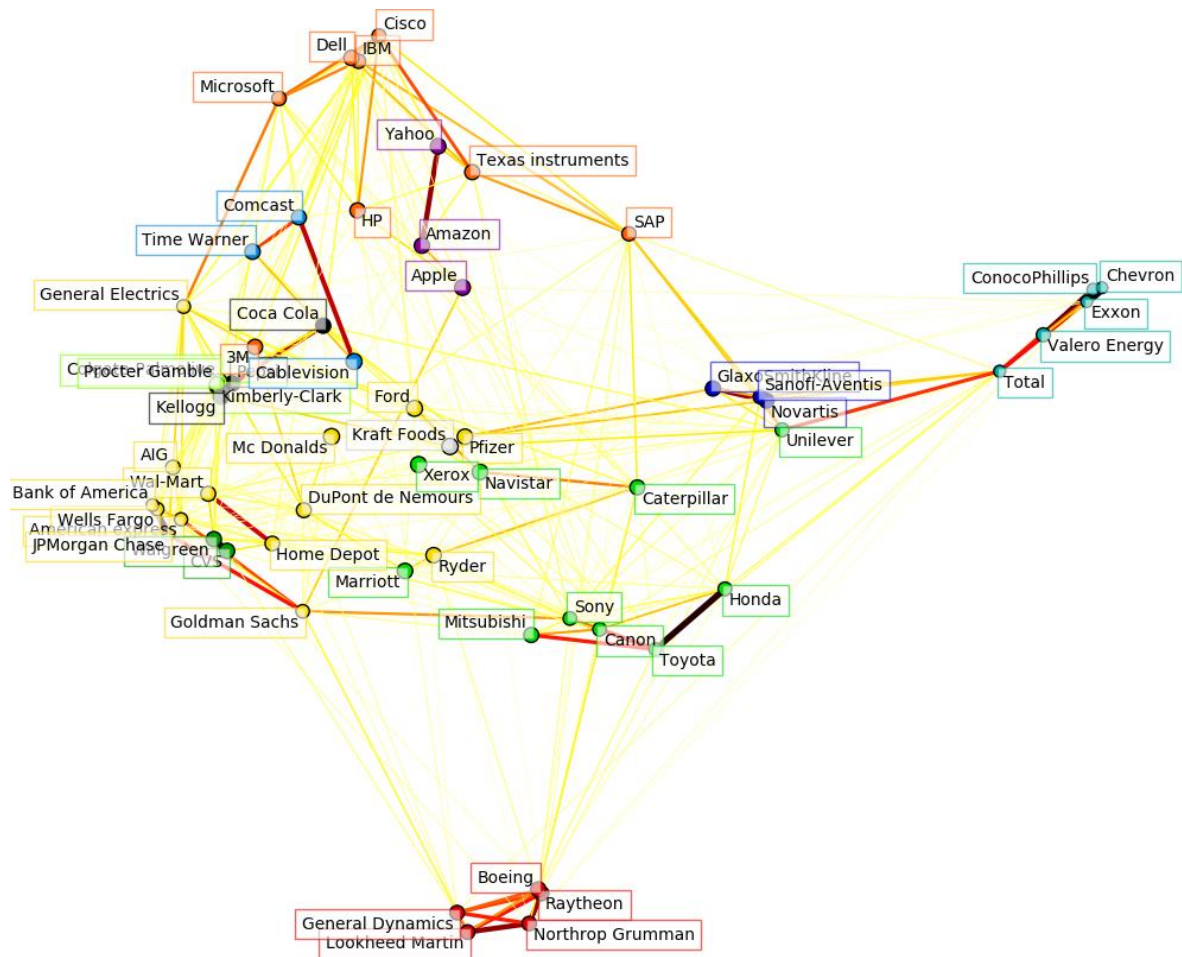
Due date : Next Week

Bottom Line : 会使用api

作业2

对沪深三百股票进行聚类
并画出右图所示的二维嵌入

Hint :



http://scikit-learn.org/stable/auto_examples/applications/plot_stock_market.html#stock-market

时间序列分析

- Mean Reversion and Ornstein-Uhlenbeck process

$$dx_t = \theta(\mu - x_t)dt + \sigma dW_t$$

ADF Test

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \cdots + \delta_{p-1} \Delta y_{t-p+1} + \epsilon_t$$

- Calculate the *test statistic*, DF_τ , which is used in the decision to reject the null hypothesis
- Use the *distribution* of the test statistic (calculated by Dickey and Fuller), along with the critical values, in order to decide whether to reject the null hypothesis

ADF Test

```
(0.049177575166452235,  
0.96241494632563063,  
1,  
3771,  
{ '1%': -3.4320852842548395,  
  '10%': -2.5671781529820348,  
  '5%': -2.8623067530084247},  
19576.116041473877)
```

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \cdots + \delta_{p-1} \Delta y_{t-p+1} + \epsilon_t$$

```
# Download the Amazon OHLCV data from 1/1/2000 to 1/1/2015  
amzn = web.DataReader("AMZN", "yahoo", datetime(2000,1,1), datetime(2015,1,1))  
# Output the results of the Augmented Dickey-Fuller test for Amazon  
# with a lag order value of 1  
ts.adfuller(amzn['Adj Close'], 1)
```

Hurst Exponent

$$\langle |\log(t + \tau) - \log(t)|^2 \rangle \sim \tau^{2H}$$

- $H < 0.5$ - The time series is mean reverting
- $H = 0.5$ - The time series is a Geometric Brownian Motion
- $H > 0.5$ - The time series is trending


```
def hurst(ts):  
    """Returns the Hurst Exponent of the time series vector ts"""  
    # Create the range of lag values  
    lags = range(2, 100)  
  
    # Calculate the array of the variances of the lagged differences  
    tau = [sqrt(std(subtract(ts[lag:], ts[: -lag]))) for lag in lags]  
  
    # Use a linear fit to estimate the Hurst Exponent  
    poly = polyfit(log(lags), log(tau), 1)  
  
    # Return the Hurst exponent from the polyfit output  
    return poly[0]*2.0
```

```
# Create a Gometric Brownian Motion, Mean-Reverting and Trending Series
gbm = log(cumsum(randn(100000))+1000)
mr = log(randn(100000)+1000)
tr = log(cumsum(randn(100000)+1)+1000)

# Output the Hurst Exponent for each of the above series
# and the price of Amazon (the Adjusted Close price) for
# the ADF test given above in the article
print("Hurst(GBM):    %s" % hurst(gbm))
print("Hurst(MR):     %s" % hurst(mr))
print("Hurst(TR):     %s" % hurst(tr))

# Assuming you have run the above code to obtain 'amzn'!
print("Hurst(AMZN):   %s" % hurst(amzn['Adj Close']))
```

```
Hurst(GBM):    0.502051910931
Hurst(MR):     0.000166110248967
Hurst(TR):     0.957701001252
Hurst(AMZN):   0.454337476553
```

作业三

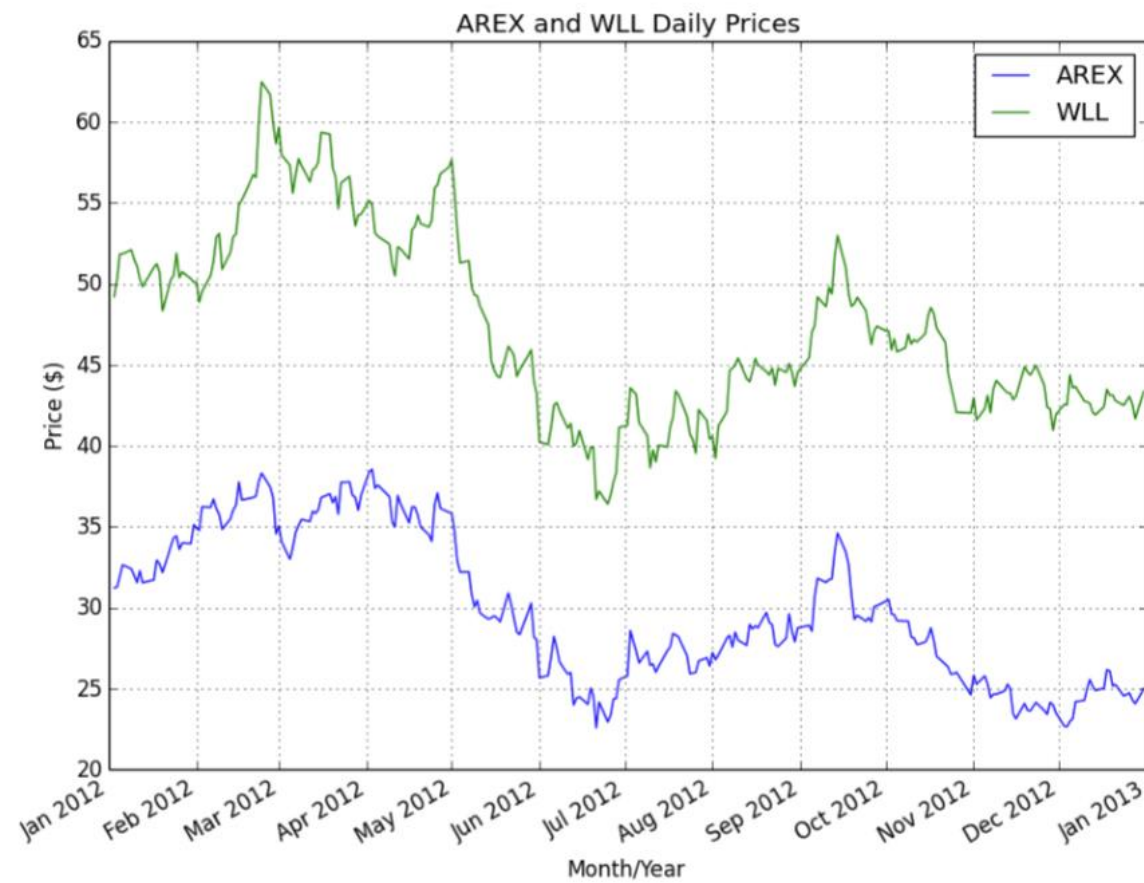
分别对沪深三百股票进行上述两者统计检测（2010-2015），并汇报出具备mean-reverting 的股票（如果存在的话）

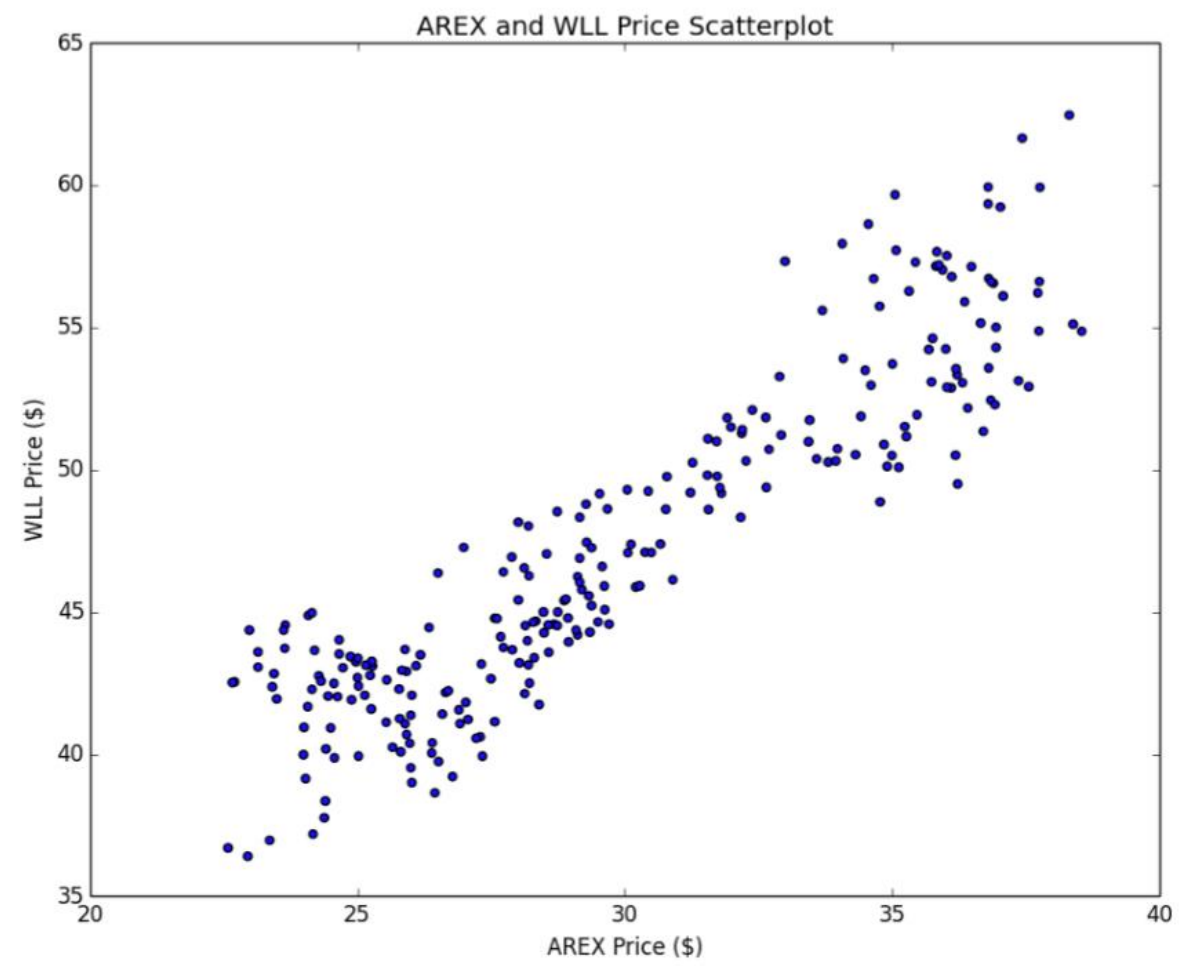
从单一股票到投资组合

- Cointegrated Augmented Dickey-Fuller Test

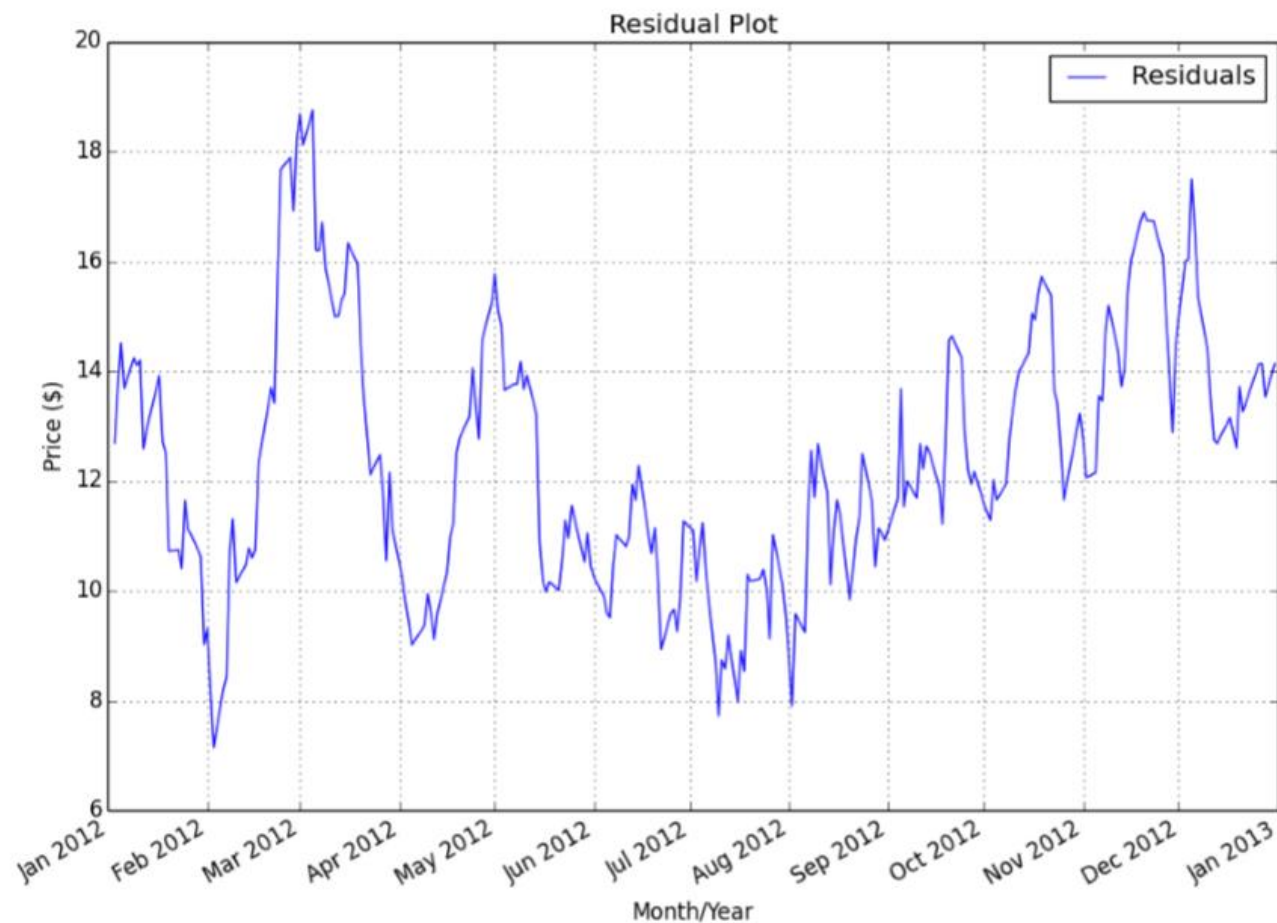
$$y(t) = \beta x(t) + \epsilon(t)$$

- 课件
- Cadf.py





```
(-2.9607012342275936,  
0.038730981052330332,  
0,  
249,  
{ '1%': -3.4568881317725864,  
  '10%': -2.5729936189738876,  
  '5%': -2.8732185133016057},  
601.96849256295991)
```



作业四

撰写脚本，提取沪深三百股票中具备上述协整关系的pairs

Machine learning: A different Approach

- $Y = f(x) + e$
- Logistic regression
- SVM
- Random Forest
- LSTM
- ...and more