금융 데이터 이해와 분석

(Finance Data in Python)

이승준 (PyCon 2014 Korea)

fb.com/plusjune

목차

목차

1. 파이썬스 파이썬 도구들

금융 데이터 분석을 위한 파이썬 도구들

데이터 분석

- IPython
- NumPy
- pandas

스크래핑

- BeautifulSoup
- StringIO
- requests

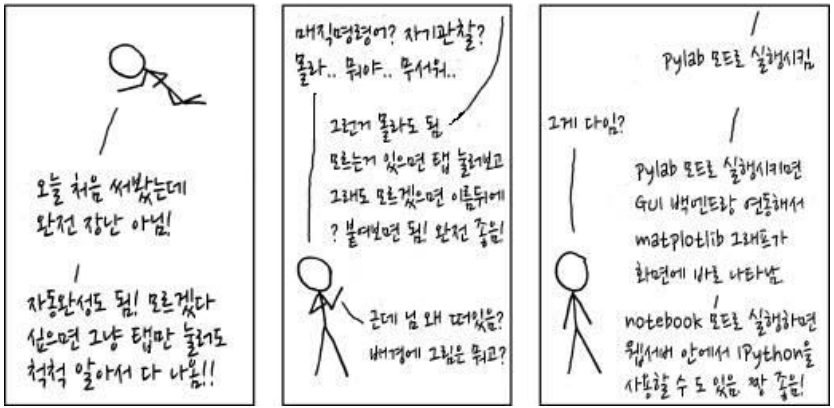
시각화 도구

- Matplotlib

IPython

브라우저 기반 인터랙티브 파이썬 환경

- 인터랙티브 파이썬 코딩
- 매직 명령어
- 시스템 쉘
- 마크다운 문법
- 수학식 표현 (Latex)
- 슬라이드, HTML, PDF
- 대규모 병렬 컴퓨팅 (IPython.parallel)



NumPy

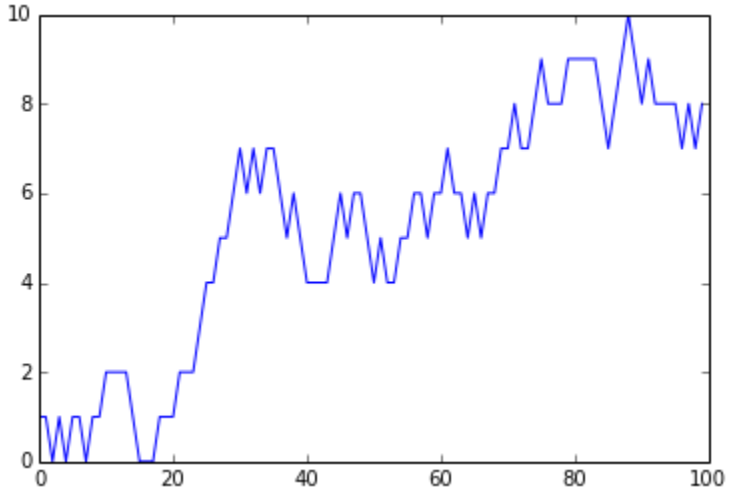
- 수치 데이터를 다루는 어마무시한 방법과 성능을 제공
- 강력한 다차원 배열(ndarry)
- 선형대수, 푸리에 변환, 랜덤 넘버 기능 등
- pandas 가 NumPy 기반

랜덤 워크 시뮬레이션

```
import numpy as np

t = np.random.randint(-1, 2, size=100) # 이전 가격대비 -1, 0, 1 (상승, 동일, 하락)
walk = t.cumsum()
walk
```

```
array([ 1,  1,  0,  1,  0,  1,  1,  0,  1,  1,  2,  2,  2,  2,  1,  0,  0,
        0,  1,  1,  1,  2,  2,  2,  3,  4,  4,  5,  5,  6,  7,  6,  7,  6,
        7,  7,  6,  5,  6,  5,  4,  4,  4,  4,  5,  6,  5,  6,  6,  5,  4,
        5,  4,  4,  5,  5,  6,  6,  5,  6,  6,  7,  6,  6,  5,  6,  5,  6,
        6,  7,  7,  8,  7,  7,  8,  9,  8,  8,  8,  9,  9,  9,  9,  9,  8,
        7,  8,  9, 10,  9,  8,  9,  8,  8,  8,  8,  7,  8,  7,  8])
```

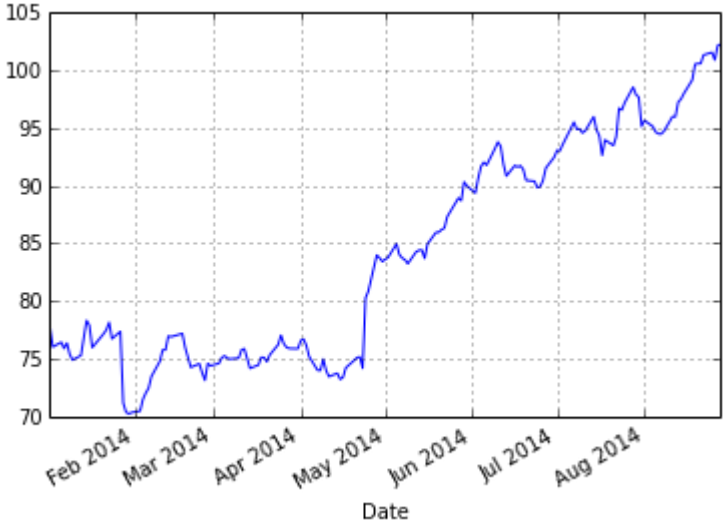
```
import matplotlib.pylab as plt
plt.plot(walk)
```



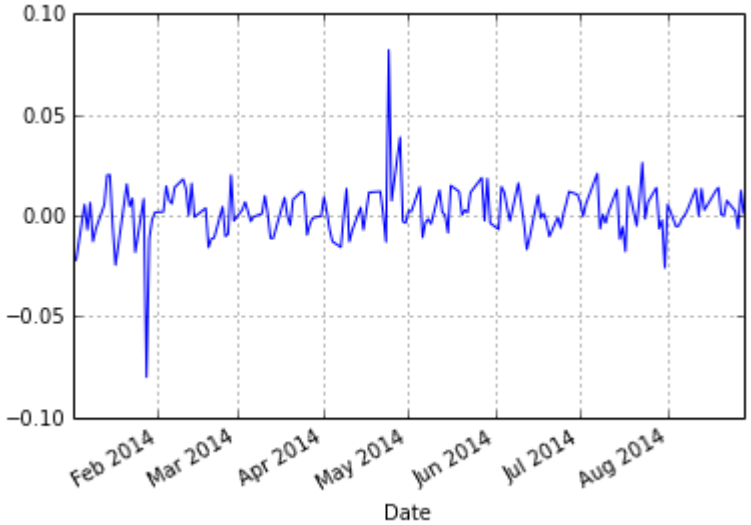
pandas

- panel data analysis 다차원 구조화된 데이터 (계량경제학)
- 데이터 분석 라이브러리 (데이터 분석, 클리닝, 모델링 등)
- 시작(2009년)은 금융 데이터 분석을 위해 설계
- 시계열 데이터 기능 통합
- R 혹은 Matlab 사용자들이 접근하기 용이

```
import pandas as pd
from pandas.io.data import DataReader

aapl = DataReader('AAPL', 'yahoo', start='2014')
aapl['Adj Close'].plot()
```



```
returns = aapl['Adj Close'].pct_change()
returns.plot()
```



2. 마켓 분석과 마켓 데이터

증권 분석은 크게 3가지로 나뉩니다

1. 기본 분석 Fundamental analysis

기본요인(fundamentals)과 내재가치(intrinsic value)를 분석

2. 기술 분석 Technical analysis

과거 가격과 거래량의 변화를 기초로 가격의 움직임을 예측

3. 정서 분석 Sentiment analysis

투자자들의 시장에 대한 욕심이나 두려움을 분석

기본 분석 Fundamental analysis

시장가치와 내재가치와 괴리가 생기고,

주가(시장가격)가 내재가치에 수렴한다는 전제

시장가치와 내재가치

- 재무제표, 주가와 손익, 산업자료 등 수치 자료 - 양적분석
- 경기, 산업동향, 노사문제, CEO의 능력 등 - 질적분석
- 기술 분석을 위해서도 기본 분석은 중요

기본 분석의 한계

- 재무제표의 데이터는 시의성이 떨어진다 (년간 4회)
- 시장가치와 내재가치와 괴리가 지나치게 오래 지속될 수 있다
- 내재가치 평가에 주관적 판단 (검증이 어렵다)

기술 분석 Technical analysis

과거 데이터를 근거로, 추세 예측.

차트를 들여다 보고 있다면, 기술 분석을 하고 있다고 보면 된다.

기술 분석의 가정

- 가격에는 모든 정보가 반영되어 있다.
- 가격은 일정한 추세로 움직인다.
- 가격의 움직임은 반복된다.

기술 분석 비판(한계)

- 예측 가능성에 대해서는 논란 (그럼에도 불구하고 "추세"를 파악하는 도구로 사용)
- 수치와 차트를 사용하기 때문에 객관적인 것 처럼 보이지만 해석이 주관적.
- 한계와 단점을 이해 할 필요

정서 분석 Sentiment analysis, Opinion mining

(자연어 처리, 텍스트 분석 등을 통해) 주관적인 정보를 추출

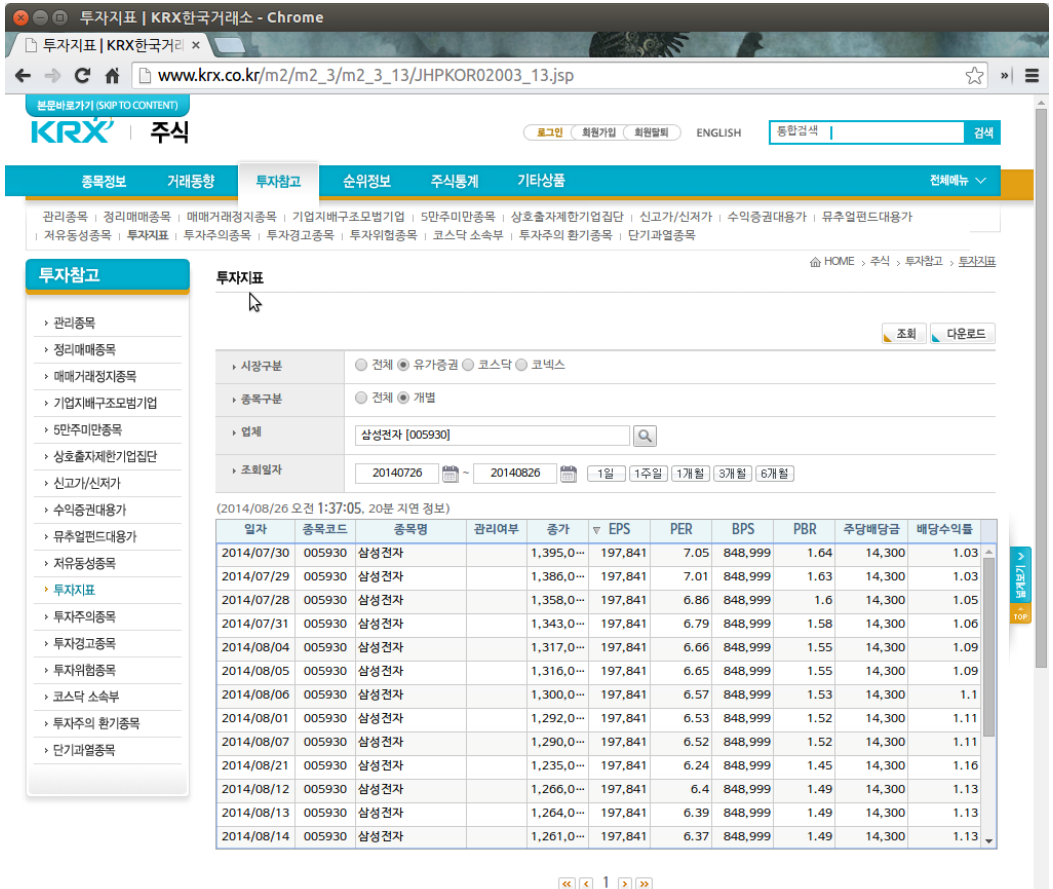
정서 분석 이란?

- 대상에 대한 (긍정적/부정적) 표현을 분석하여 의사결정에 활용
- 투자자들의 두려움이나 욕심을 수치화 활용

정서 분석 관련 연구

- 텍스트 마이닝
- 뉴스 텍스트 마이닝
- 어휘 트렌드 분석
- 소셜 정서 분석

한국거래소 주식 투자지표



일자	종목코드	종목명	관리여부	종가	EPS	PER	BPS	PBR	주당배당금	배당수익률
2014/07/30	005930	삼성전자		1,395,0…	197,841	7.05	848,999	1.64	14,300	1.03
2014/07/29	005930	삼성전자		1,386,0…	197,841	7.01	848,999	1.63	14,300	1.03
2014/07/28	005930	삼성전자		1,358,0…	197,841	6.86	848,999	1.6	14,300	1.05
2014/07/31	005930	삼성전자		1,343,0…	197,841	6.79	848,999	1.58	14,300	1.06
2014/08/04	005930	삼성전자		1,317,0…	197,841	6.66	848,999	1.55	14,300	1.09
2014/08/05	005930	삼성전자		1,316,0…	197,841	6.65	848,999	1.55	14,300	1.09
2014/08/06	005930	삼성전자		1,300,0…	197,841	6.57	848,999	1.53	14,300	1.1
2014/08/01	005930	삼성전자		1,292,0…	197,841	6.53	848,999	1.52	14,300	1.11
2014/08/07	005930	삼성전자		1,290,0…	197,841	6.52	848,999	1.52	14,300	1.11
2014/08/21	005930	삼성전자		1,235,0…	197,841	6.24	848,999	1.45	14,300	1.16
2014/08/12	005930	삼성전자		1,266,0…	197,841	6.4	848,999	1.49	14,300	1.13
2014/08/13	005930	삼성전자		1,264,0…	197,841	6.39	848,999	1.49	14,300	1.13
2014/08/14	005930	삼성전자		1,261,0…	197,841	6.37	848,999	1.49	14,300	1.13

EPS,
PER,
BPS,
PBR,
주당배당금,
배당수익률

주요 투자지표

- PER : 주가 / 주당순익(EPS)
- EPS : 순이익 / 주식수
- PBR : 주가 / 주당순자산(BPS)
- BPS : 순자산 / 주식수

마켓 데이터 소스 5 (로그인도 필요 없는)

1. **KRX 한국거래소** http://krx.co.kr
   증권 시장 전체 데이터, 증권 통계, 표준코드 시스템

2. **전자공시시스템(금융감독원) DART** http://dart.fss.or.kr
   기업 공시, 재무제표

3. 포탈 증권 (네이버, Daum) http://stock.naver.com
   종목 정보 상세, 뉴스, 투자 정보

마켓 데이터 소스 5 (로그인도 필요 없는)

4. **야후 파이낸스** http://finance.yahoo.com
   세계/미국 증시, 뉴스, 종목 정보

5. **연방준비은행 경제 데이터** http://research.stlouisfed.org/fred2
   거시 분석, 해외 시황 (포괄적인 경제 데이터)

실시간 데이터, 정말 중요한가

트레이딩 스타일에 따라 다르다

Trading Style	매매 간격	보유기간	비고
Position Trading	장기간	수 개월~ 수 년	-
Swing Trading	짧은 기간	몇 일~ 몇 주	-
Day Trading	수 시간	하루는 넘기지 않음	오버나잇 포지션 없음
Scalp Trading	초단타	수초~ 수분 단위로 수십번 거래	오버나잇 포지션 없음

주요 수집 대상 데이터

- 종목 코드 (심볼 목록)
- 종목 기본 정보, 상태
- 재무정보
- 주가 데이터 (틱,초,분,일,주,월) - 틱,분,일
- 뉴스, SNS 텍스트 등 비정형 데이터

거래소 홈페이지 예

- 일자별 주가: 전체, 과거 10년치 이상
- 뉴스/공시: 공시 전체, 뉴스(과거 6개월)
- 투자자별 거래실적 (일일, 과거 10년치 이상)
- 기관외국인동향: 일자별 매수,매도

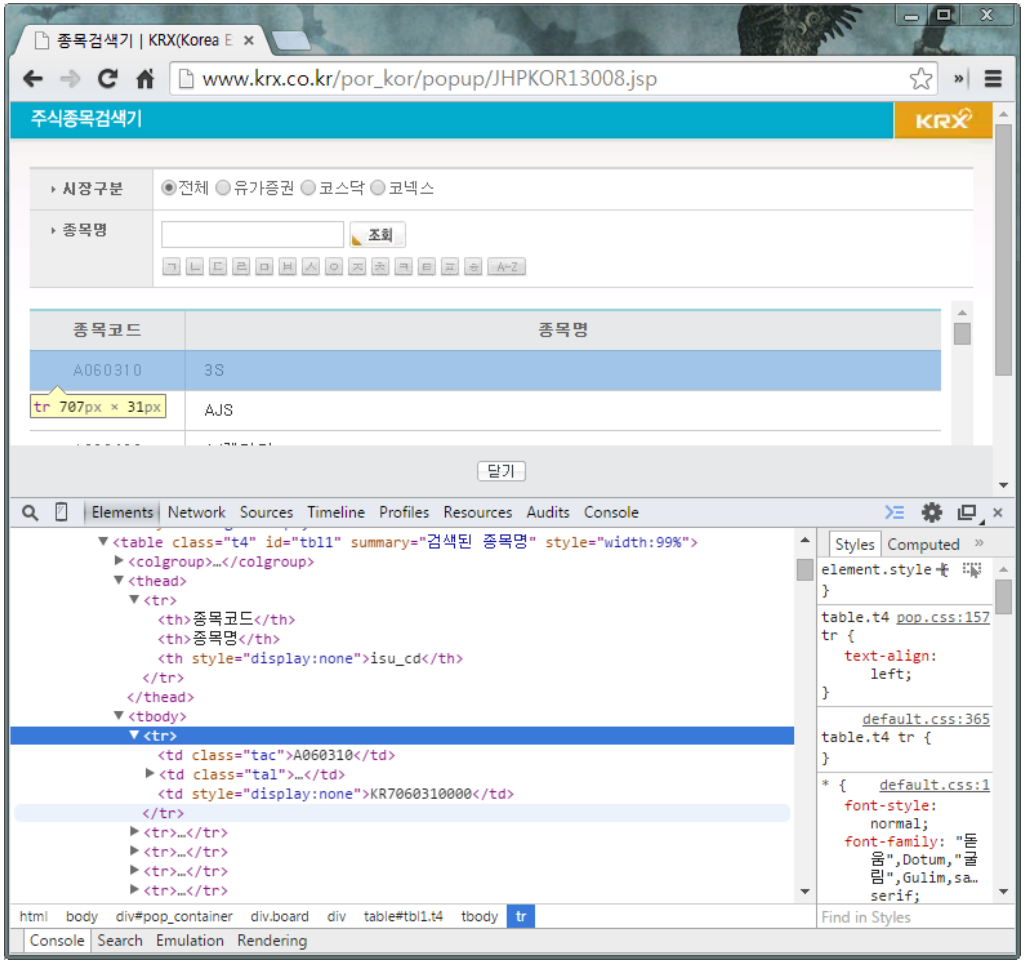
3. 주식 종목 코드

거래소 주식종목검색기 http://krx.co.kr/por_kor/popup/JHPKOR13008.jsp

Chrome 개발자 도구

HTML DOM 구조 파악
(Table, Xpath)

//*[@id="tbl1"]



BeautifulSoup으로 HTML 파싱

//*[@id="tbl1"]

```
from BeautifulSoup import BeautifulSoup

html_text = "<html lang="ko" dir="ltr" class="client-js"><head>… "

soup = BeautifulSoup(html_text)
table = soup.find('table', {'id':'tbl1'})
trs = table.findAll('tr')
```

전체 종목코드를 JSON으로 저장

http://www.krx.co.kr/por_kor/popup/JHPKOR13008.jsp 조회 페이지

1. requests HTML 문서 받기
2. BeautifulSoup HTML 문서 파싱
3. 데이터 추출하여 리스트로 구성
4. 리스트를 JSON 포맷으로 저장

#1 . requests HTML 문서 받기

```
import requests
import json
from BeautifulSoup import BeautifulSoup

url = 'http://www.krx.co.kr/por_kor/popup/JHPKOR13008.jsp'
r = requests.post(url, data={'mkt_typ':'S', 'market_gubun': 'allVal'})
```

#2. BeautifulSoup HTML 분석 파싱!

```
soup = BeautifulSoup(r.text)
table = soup.find('table', {'id':'tbl1'})
trs = table.findAll('tr')
```

#3. 데이터 추출하여 리스트로 구성

```
stock_list = []

for tr in trs[1:]:
    stock = {}
    cols = tr.findAll('td')
    stock['code'] = cols[0].text[1:]
    stock['name'] = cols[1].text.replace(";", "")
    stock['full_code'] = cols[2].text
    stock_list.append(stock)
```

#4. 리스트를 JSON 포맷으로 저장

```
j = json.dumps(stock_list)
with open('data/krx_symbols.json', 'w') as f:
    f.write(j)
```

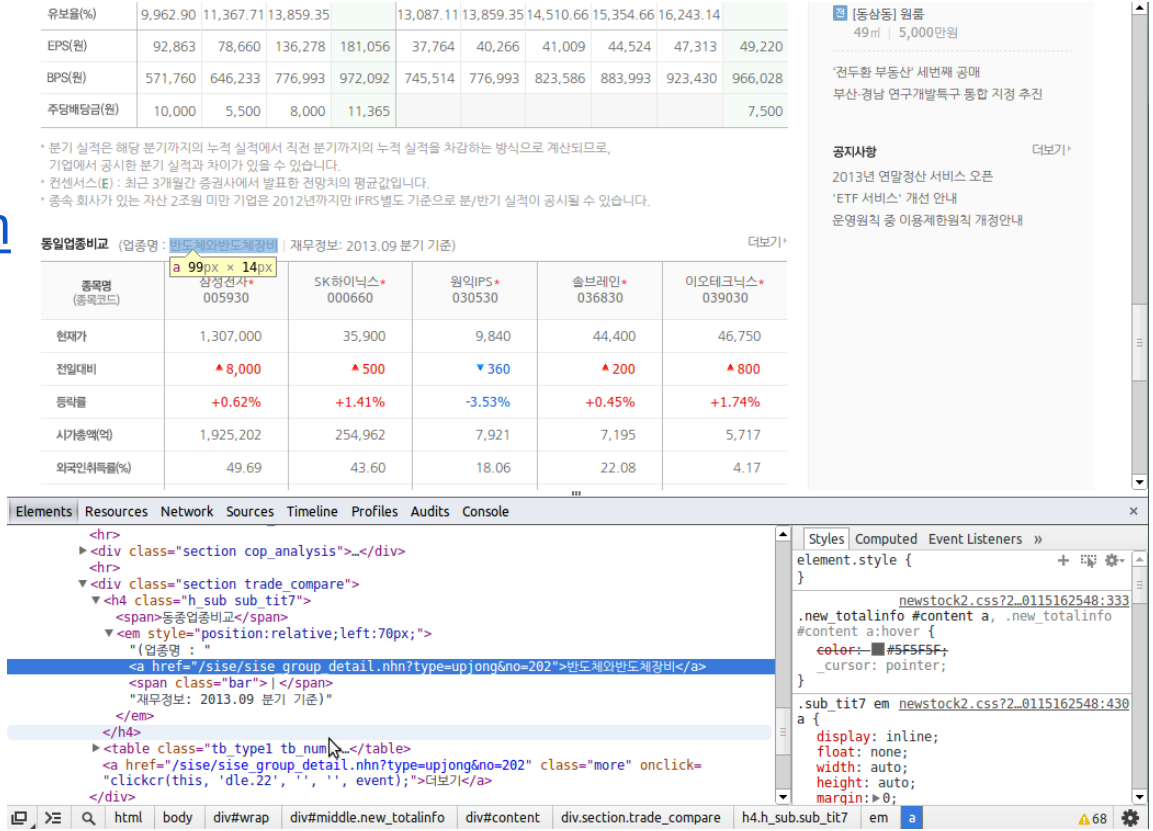
저장 내용 확인

```
fn = 'data/krx_symbols.json'
with open(fn, 'r') as f:
    stock_list = json.load(f)

for s in stock_list[:10]:
    print s['full_code'], s['code'][1:], s['name']
```

```
KR7060310000 60310 3S
KR7013340005 13340 AJS
KR7068400001 68400 AJ렌터카
KR7006840003 06840 AK홀딩스
KR7054620000 54620 AP시스템
... ...
```

업종 가져오기

http://finance.naver.com



h4 = soup.find('h4', {'class':'h_sub sub_tit7'})

```
import requests

def get_sector(code):
    url = 'http://finance.naver.com/item/main.nhn?code=' + code
    r = requests.get(url)
    soup = BeautifulSoup(r.text)

    sector = ""
    h4 = soup.find('h4', {'class':'h_sub sub_tit7'})
    if h4 is not None:
        sector = h4.a.text
    return sector

print get_sector('005930')
```

반도체와반도체장비

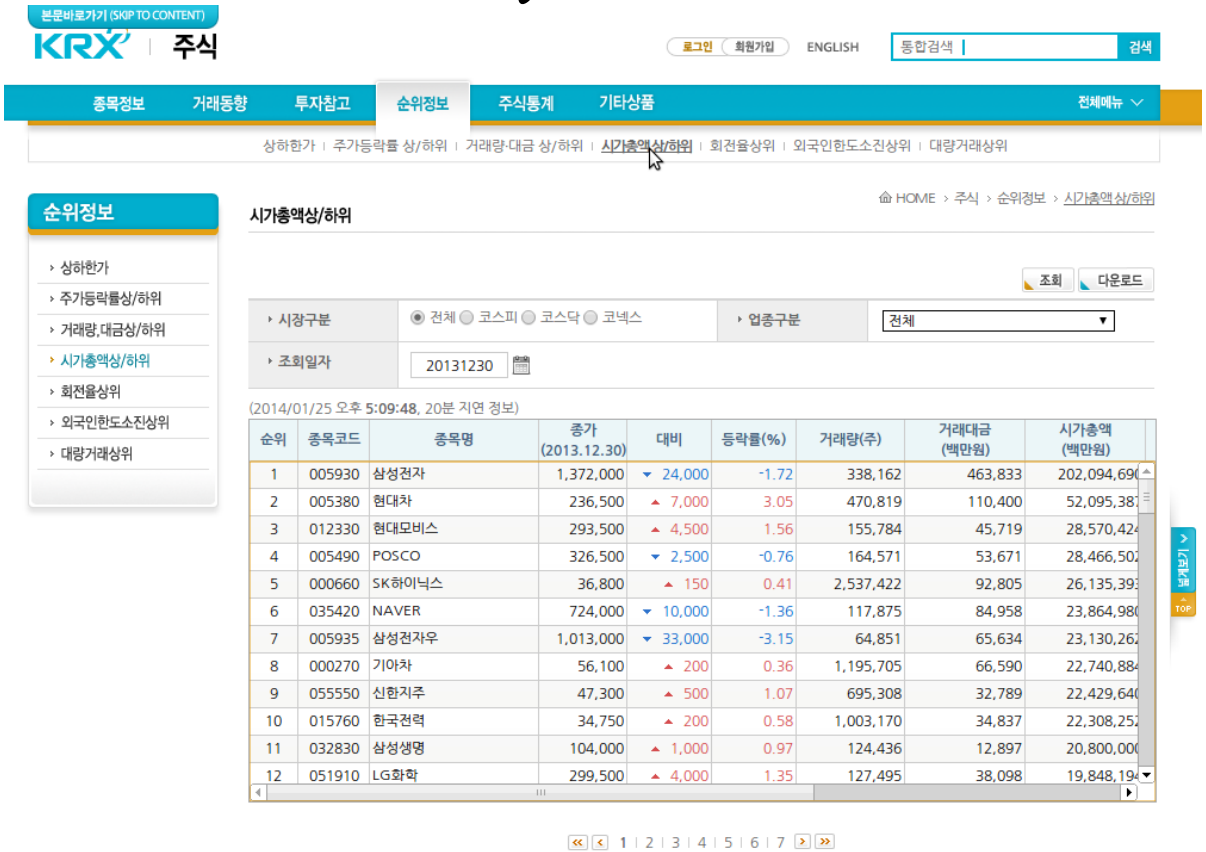
pandas.DataFrame 으로 읽기

```
import pandas as pd

fn = 'data/krx_symbols_sector.json'
df = pd.read_json(fn)
df.head()
```

	code	full_code	name	sector
	060310	KR7060310000	3S	반도체와반도체장비
	013340	KR7013340005	AJS	기계
	068400	KR7068400001	AJ렌터카	도로와철도운송
	006840	KR7006840003	AK홀딩스	화학
	054620	KR7054620000	AP시스템	디스플레이장비및부품

4. 시가총액 분석

거래소 시가총액 순위 페이지



데이터

- 구글 드라이브 스프레드시트에 저장
- 데이터 URL (1995년 ~ 2013년, 19개)
- 파일당 약 1700~2000 행(row)

1995 ~ 2005

구글 스프레드 문서	URL	CSV
marcap-1995	http://goo.gl/yqbIX9	http://goo.gl/Wn6ju9
marcap-1996	http://goo.gl/E47Fq4	http://goo.gl/gRB9pu
marcap-1997	http://goo.gl/ahjwNk	http://goo.gl/Yp84gB
marcap-1998	http://goo.gl/ZwKmro	http://goo.gl/FtMt5b
marcap-1999	http://goo.gl/wO3NZH	http://goo.gl/LTS1Xf
marcap-2000	http://goo.gl/9pvOZ9	http://goo.gl/I0RBEn
marcap-2001	http://goo.gl/vDGxkz	http://goo.gl/dPHn1e
marcap-2002	http://goo.gl/mZoJz7	http://goo.gl/VYNLe7
marcap-2003	http://goo.gl/cxzJrd	http://goo.gl/EwukOZ
marcap-2004	http://goo.gl/mmfYz6	http://goo.gl/jHS0I2
marcap-2005	http://goo.gl/8FVzwj	http://goo.gl/5qhhdT

2006 ~ 2013

구글 스프레드 문서	URL	CSV
marcap-2006	http://goo.gl/NxJU6q	http://goo.gl/EiwG7n
marcap-2007	http://goo.gl/JJ6ao3	http://goo.gl/s818MK
marcap-2008	http://goo.gl/xE3uaN	http://goo.gl/Mq6Vut
marcap-2009	http://goo.gl/xqTHFZ	http://goo.gl/8kNCZW
marcap-2010	http://goo.gl/KkGU2j	http://goo.gl/w2ugn1
marcap-2011	http://goo.gl/Ff8hsr	http://goo.gl/x6mQIN
marcap-2012	http://goo.gl/S1CS26	http://goo.gl/Mf1KFu
marcap-2013	http://goo.gl/4PtBW4	http://goo.gl/WfQnX0

데이터 파일 구성

marcap-2013.csv (2013년 시총 순위)

	rank	code	name	marcap	marcap_pct	year	sector
0	1	005930	삼성전자	202.095	15.47%	2013	반도체와반도체장비
1	2	005380	현대차	52.095	3.99%	2013	자동차
2	3	012330	현대모비스	28.570	2.19%	2013	자동차부품
3	4	005490	POSCO	28.467	2.18%	2013	철강
4	5	000660	SK하이닉스	26.135	2.00%	2013	반도체와반도체장비
5	6	035420	NAVER	23.865	1.83%	2013	인터넷소프트웨어와서비스
6	7	005935	삼성전자우	23.130	1.77%	2013	반도체와반도체장비
7	8	000270	기아차	22.741	1.74%	2013	자동차
8	9	055550	신한지주	22.430	1.72%	2013	은행
...	...	...	...	...	...	...	...

데이터 사용

- requests 모듈 사용
- goo.gl 의 shorten URL 을 이용해도 사용이 가능
- csv 형태 문자열로 받아 StringIO 를 통해 읽는다

사용예

```
import pandas as pd
import requests
from StringIO import StringIO

# 2013년 시가총액
url = "https://docs.google.com/spreadsheet/ccc?" +
      "key=0Auils-M1uCmvdGFSZFVual9aYVp1UE9oWnZocE5aQVE" +
      "&output=csv"

url = "http://goo.gl/WfQnX0" # csv 로 다운로드

r = requests.get(url)
data = r.content
```

```
df = pd.read_csv(StringIO(data), dtype={'code':np.str})
df.head()
```

	Unnamed: 0	rank	code	name	marcap	marcap_pct	year	sector
0	0	1	005930	삼성전자	202.095	15.47%	2013	반도체와반도체장비
1	1	2	005380	현대차	52.095	3.99%	2013	자동차
2	2	3	012330	현대모비스	28.570	2.19%	2013	자동차부품
3	3	4	005490	POSCO	28.467	2.18%	2013	철강
4	4	5	000660	SK하이닉스	26.135	2.00%	2013	반도체와반도체장비

필요한 컬럼만 추출

```
df2 = df[['code', 'name', 'marcap', 'sector']]
df2.head(10)
```

	code	name	marcap	sector
0	005930	삼성전자	202.095	반도체와반도체장비
1	005380	현대차	52.095	자동차
2	012330	현대모비스	28.570	자동차부품
3	005490	POSCO	28.467	철강
4	000660	SK하이닉스	26.135	반도체와반도체장비
5	035420	NAVER	23.865	인터넷소프트웨어와서비스
.	…	…	…	…

개수를 구하기

- Series.value_counts() 는 고유한 값의 개수를 구하는 간편한 방법을 제공
- 업종(sector)의 고유한 값의 개수

```
sector_counts = df2['sector'].value_counts()

print sector_counts.count() # 업종수: 80
print sector_counts.index # 업종항목: 자동차부품, 화학, 기계, 제약 등 …
print sector_counts.values # 업종내 기업수: 120, 109, 98, 95 …
```

업종별로 색상 테이블 (색상 테이블 해쉬)

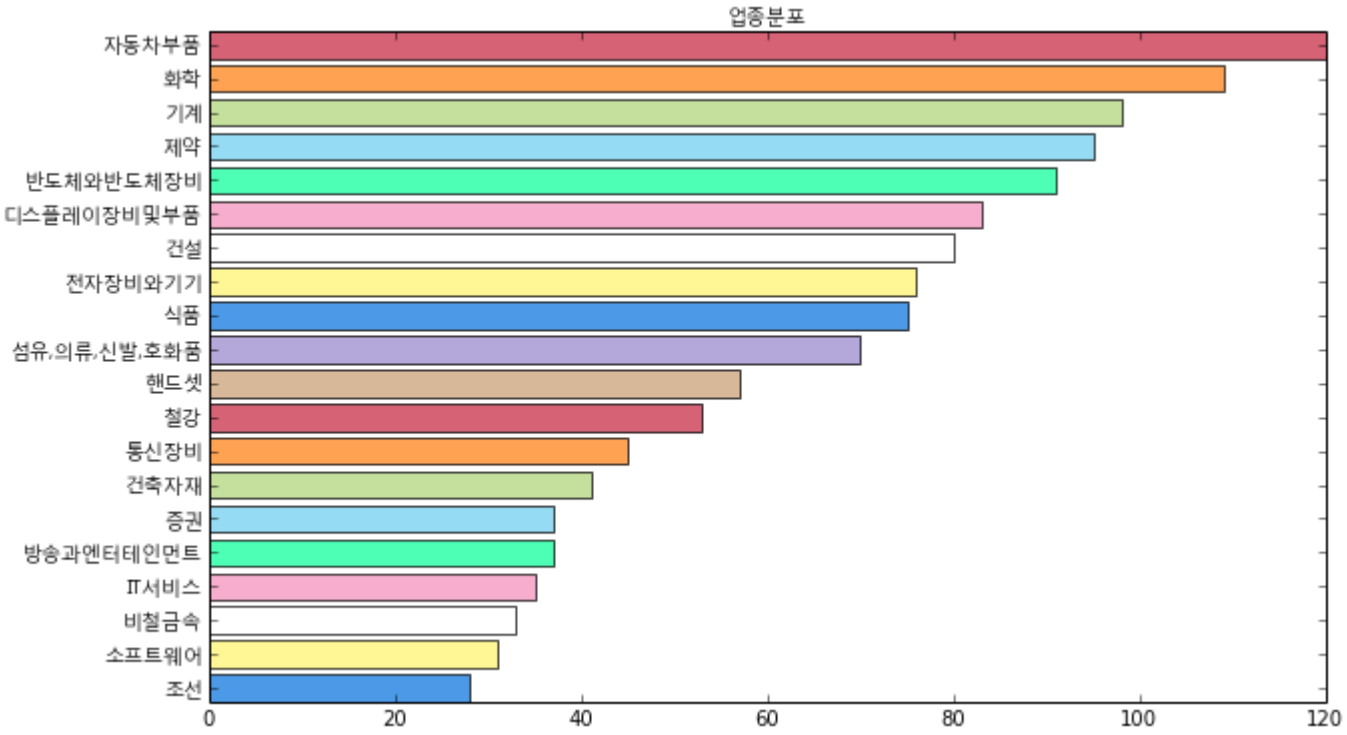
```
from itertools import cycle

colors_list = [ "#C41F3B", "#FF7D0A", "#ABD473", "#69CCF0", "#00FF96",
"#F58CBA", "#FFFFFF", "#FFF569", "#0070DE", "#9482C9", "#C79C6E" ]
color = cycle(colors_list)

print next(color)
print next(color)
print next(color)
```

#C41F3B
#FF7D0A
#ABD473

업종 분포 차트

```
import matplotlib.pylab as plt
import matplotlib.font_manager as fm
fontprop = fm.FontProperties(fname="fonts/malgun.ttf")

top20 = sector_counts[0:20]

fig, ax = plt.subplots(figsize=(10, 6))
ax.set_title(u'업종분포', fontproperties=fontprop)
pos = arange(20)
pos = pos[::-1] # reverse pos list
plt.yticks(pos, [x.decode('utf8') for x in top20.index], fontproperties=fontprop)
plt.barh(pos, top20.values, align='center', color=colors_list, alpha=0.7)
```

업종 분포 차트


업종분포
자동차부품
화학
기계
제약
반도체와반도체장비
디스플레이장비및부품
건설
전자장비와기기
식품
섬유,의류,신발,호화품
핸드셋
철강
통신장비
건축자재
증권
방송과엔터테인먼트
IT서비스
비철금속
소프트웨어
조선
0
20
40
60
80
100
120

특정 섹터만 추출

```
df_semi = df[df['sector']=='소프트웨어']
df_semi.head(10)
```

	Unnamed: 0	rank	code	name	marcap	marcap_pct	year	sector
211	211	212	053800	안랩	0.617	0.05%	2013	소프트웨어
248	248	249	030520	한글과컴퓨터	0.469	0.04%	2013	소프트웨어
322	322	323	012510	더존비즈온	0.332	0.03%	2013	소프트웨어
455	455	456	053980	오상자이엘	0.205	0.02%	2013	소프트웨어
458	458	459	041020	인프라웨어	0.204	0.02%	2013	소프트웨어
476	476	477	136540	윈스테크넷	0.197	0.02%	2013	소프트웨어
492	492	493	085810	알티캐스트	0.190	0.01%	2013	소프트웨어
517	517	518	032190	다우데이타	0.179	0.01%	2013	소프트웨어
660	660	661	086960	MDS테크	0.128	0.01%	2013	소프트웨어
737	737	738	078000	텔코웨어	0.111	0.01%	2013	소프트웨어

시가총액 합산

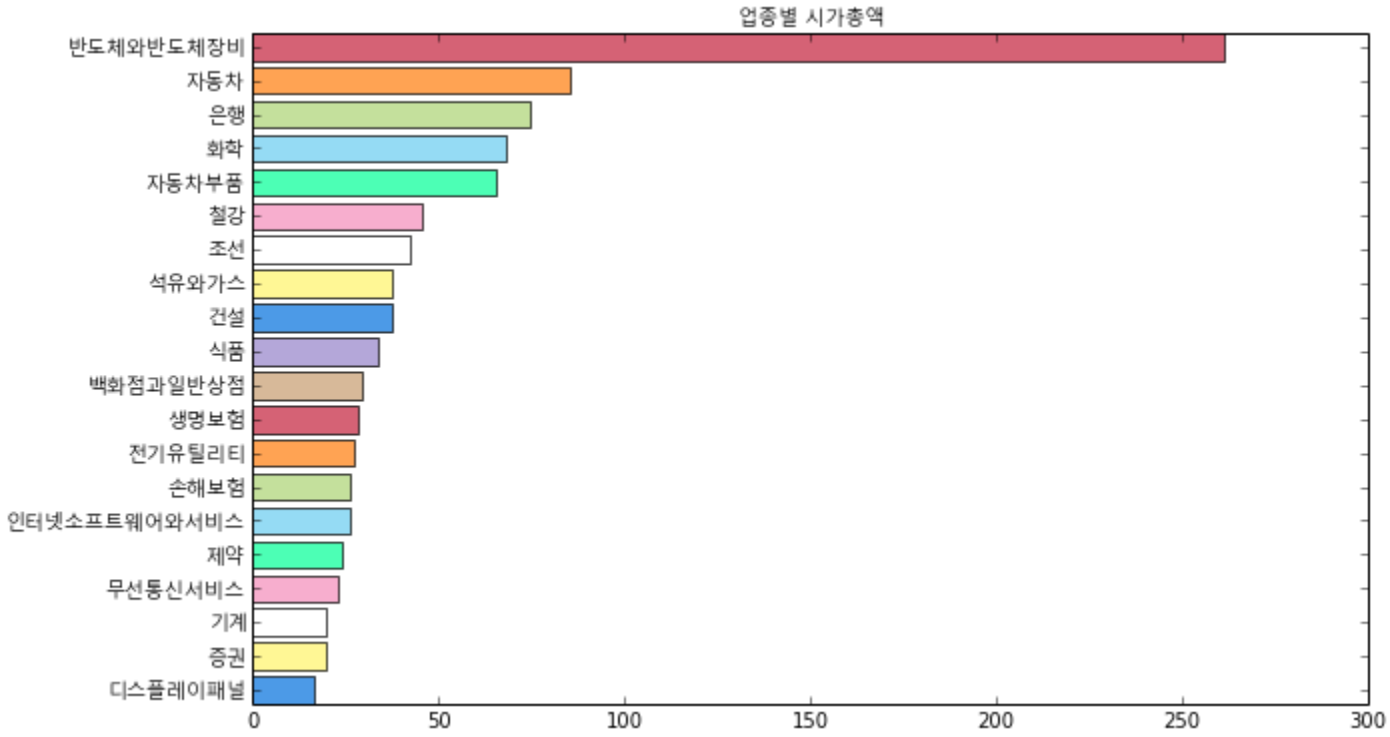
pivot_table(), 피벗 테이블

```
from pandas.tools.pivot import pivot_table

ttable = df[['sector', 'marcap']]
piv = pivot_table(ttable, values='marcap', rows=['sector'], aggfunc=np.sum)
sector_marcap = piv.copy()
sector_marcap.sort(ascending=False)
sector_marcap[:10]
```

```
sector
반도체와반도체장비    261.118
자동차          85.512
은행           74.734
화학           68.137
자동차부품         65.605
...
```

업종별 시가총액 차트

```
import matplotlib.pylab as plt
import matplotlib.font_manager as fm
fontprop = fm.FontProperties(fname="fonts/malgun.ttf")

top20 = sector_marcap[0:20]

fig, ax = plt.subplots(figsize=(10, 6))
ax.set_title(u'업종별 시가총액', fontproperties=fontprop)
pos = arange(20)
pos = pos[::-1] # reverse pos list
plt.yticks(pos, [x.decode('utf8') for x in top20.index], fontproperties=fontprop)
plt.barh(pos, top20.values, align='center', color=colors_list, alpha=0.7)
```


업종별 시가총액
반도체와반도체장비
자동차
은행
화학
자동차부품
철강
조선
석유와가스
건설
식품
백화점과일반상점
생명보험
전기유틸리티
손해보험
인터넷소프트웨어와서비스
제약
무선통신서비스
기계
증권
디스플레이패널
0
50
100
150
200
250
300

업종별 시가총액 비중

```
sector_marcap_pct = sector_marcap / sector_marcap.sum()
sector_marcap_pct[:10]
```

```
sector
반도체와반도체장비    0.201072
자동차          0.065848
은행           0.057548
화학           0.052468
자동차부품        0.050519
…
```

반도체와반도체장비 0.201072

즉, 시장 시가총액의 20% 차지

```
print sector_counts.index[:10]
print sector_counts.values[:10]

print sector_marcap.index[:10]
print sector_marcap.values[:10]
```

```
Index([u'자동차부품', u'화학', u'기계', u'제약', u'반도체와반도체장비', u'디스플레이장비
및부품', u'건설', u'전자장비와기기', u'식품', u'섬유,의류,신발,호화품'], dtype='object')
[120 109  98  95  91  83  80  76  75  70]
Index([u'반도체와반도체장비', u'자동차', u'은행', u'화학', u'자동차부품', u'철강', u'조선',
u'석유와가스', u'건설', u'식품'], dtype='object')
[ 261.118   85.512   74.734   68.137   65.605   45.68    42.511   37.585
   37.585   33.944]
```

```
sector_counts[:10]
```

```
자동차부품          120
화학                109
기계                98
제약                95
반도체와반도체장비  91
디스플레이장비및부품  83
건설                80
전자장비와기기      76
식품                75
섬유,의류,신발,호화품  70
dtype: int64
```

시가총액 비중 차트(파이)

```
import matplotlib.font_manager as fm
fontprop = fm.FontProperties(fname="fonts/malgun.ttf")

fig, axes = plt.subplots(nrows=1, ncols=2)
fig.set_size_inches(18, 8)
```

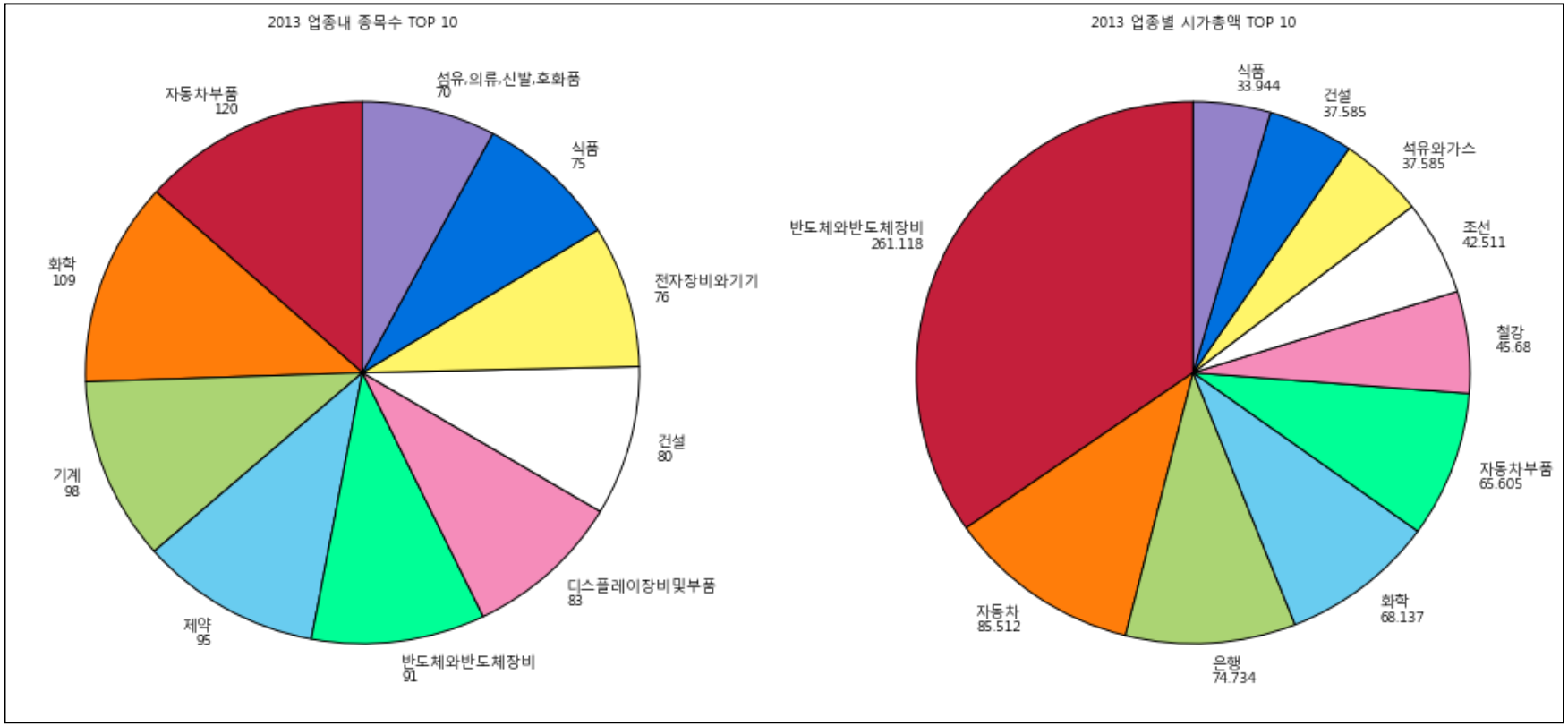
첫번째 파이 차트 (업종내 종목수)

```
sec_stock_top = sector_counts[:10]
labels = sec_stock_top.index.astype(str) + '\n' + sec_stock_top.values.astype(str)
ulabels = [x.decode('utf-8') for x in labels]

axes[0].set_title(u"2013 업종내 종목수 TOP 10", fontproperties=fontprop)
patches, texts = axes[0].pie(sec_stock_top, labels=ulabels, startangle=90,
colors=colors_list)
plt.setp(texts, fontproperties=fontprop)
```

두번째 파이 차트 (업종별 시가총액)

```
sec_mar_top = sector_marcap[:10]
labels = sec_mar_top.index.astype(str) + '\n' + sec_mar_top.values.astype(str)
ulabels = [x.decode('utf-8') for x in labels]

axes[1].set_title(u"2013 업종별 시가총액 TOP 10", fontproperties=fontprop)
patches, texts = axes[1].pie(sec_mar_top, labels=ulabels, startangle=90,
colors=colors_list)
plt.setp(texts, fontproperties=fontprop)
```

시가총액 비중 차트(파이)


2013 업종내 종목수 TOP 10
섬유,의류,신발,호화품
70
식품
75
전자장비와기기
76
건설
80
디스플레이장비및부품
83
반도체와반도체장비
91
제약
95
기계
98
화학
109
자동차부품
120
2013 업종별 시가총액 TOP 10
식품
33.944
건설
37.585
석유와가스
37.585
조선
42.511
철강
45.68
자동차부품
65.605
화학
68.137
은행
74.734
자동차
85.512
반도체와반도체장비
261.118

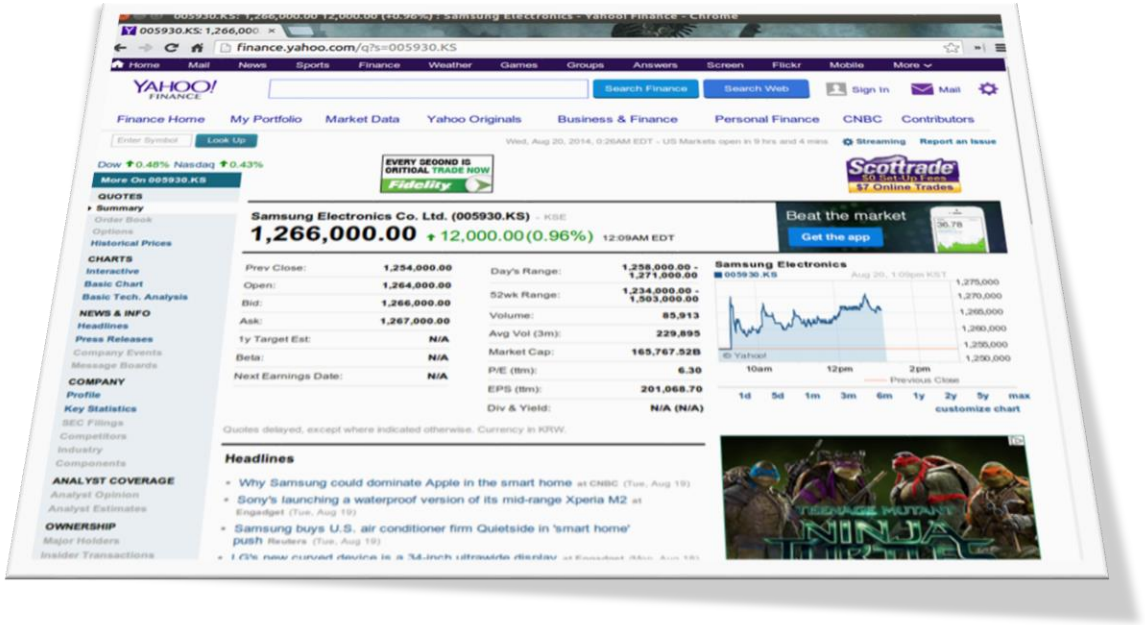
시가 총액 분석 (1990년 이후 23년간)

- 시총30위 사수 5곳 (한국전력, POSCO, 삼성전자, 현대차, LG전자)
- 한국전력(1990~1998까지 시가총액 1위)⟶삼성전자(14년째) 권력이동
- 역대 시총 2위는 POSCO, SK텔레콤, 현대차 등 소수 종목들이 번갈아
- 최상위 종목을 제외하고는 30위권 내 순위변동 극심

5. 야후 파이낸스

야후 파이낸스

http://finance.yahoo.com/q?s=005930.KS



파이넌스 심볼 (symbol)

미국시장에서는 종목을 식별하기 위해 영문알파벳 조합된 코드.
야후(yahho.com), 포브스(forbes.com) 같은 모든 경제 사이트에서 통용

뉴욕증권거래소(NYSE, 1~3개의 영문)

- **F** (포드 자동차)
- **GE** (제너럴 일렉트릭)
- **BAC** (뱅크오브아메리카)

나스닥(NASDAQ, 4~5개의 영문)

- **APPL** (Apple Inc, 애플)
- **GOOG** (Google, 구글)
- **MSFT** (Microsoft, 마이크로소프트)

모든 종목의 심볼을 구하려면,

Company List (NASDAQ, NYSE, & AMEX)

- http://nasdaq.com/screening/company-list.aspx

Nasdaqtrader.com

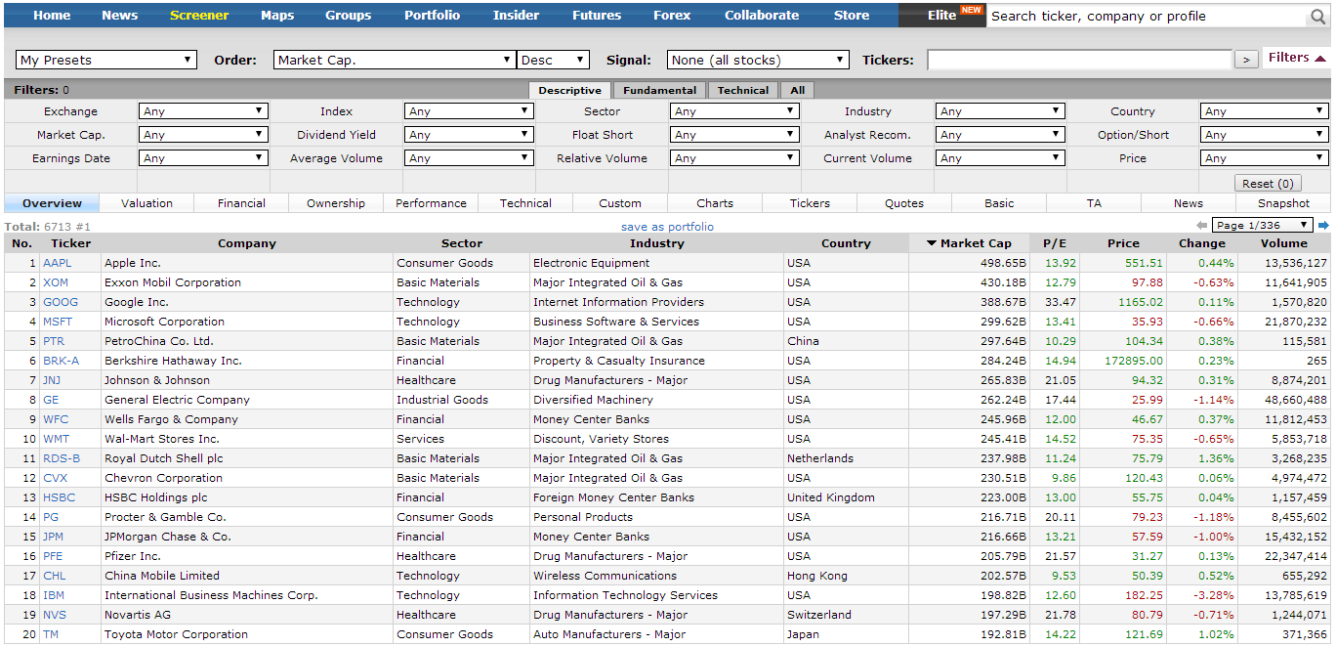
- http://www.nasdaqtrader.com/trader.aspx?id=symboldirdefs
- ftp://ftp.nasdaqtrader.com/SymbolDirectory/
- nasdaqlisted.txt, otherlisted.txt

Bloomberg Open Symbology

- http://bsym.bloomberg.com/sym/

FINVIZ.com (종목 검색기)

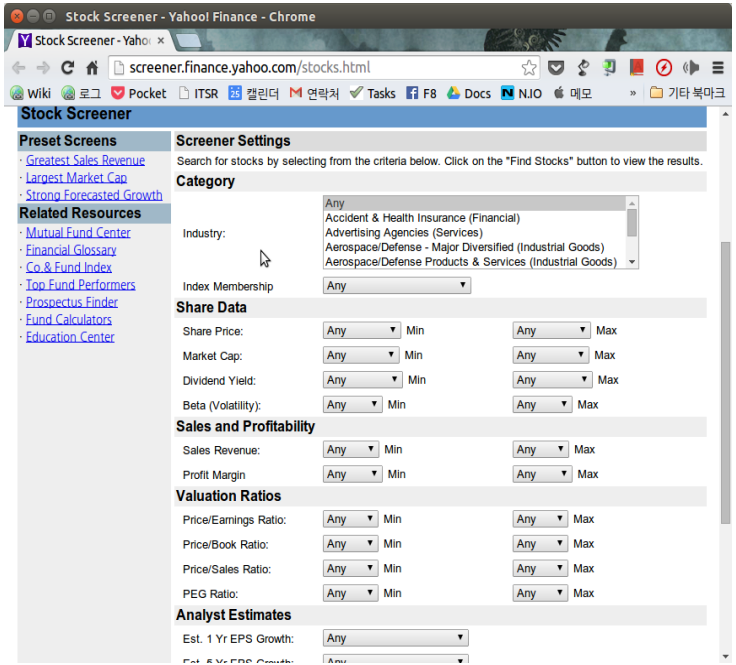
http://finviz.com/screener.ashx



No.	Ticker	Company	Sector	Industry	Country	Market Cap	P/E	Price	Change	Volume
1	AAPL	Apple Inc.	Consumer Goods	Electronic Equipment	USA	498.65B	13.92	551.51	0.44%	13,536,127
2	XOM	Exxon Mobil Corporation	Basic Materials	Major Integrated Oil & Gas	USA	430.18B	12.79	97.88	-0.63%	11,641,905
3	GOOG	Google Inc.	Technology	Internet Information Providers	USA	388.67B	33.47	1165.02	0.11%	1,570,820
4	MSFT	Microsoft Corporation	Technology	Business Software & Services	USA	299.62B	13.41	35.93	-0.66%	21,870,232
5	PTR	PetroChina Co. Ltd.	Basic Materials	Major Integrated Oil & Gas	China	297.64B	10.29	104.34	0.38%	115,581
6	BRK-A	Berkshire Hathaway Inc.	Financial	Property & Casualty Insurance	USA	284.24B	14.94	172895.00	0.23%	265
7	JNJ	Johnson & Johnson	Healthcare	Drug Manufacturers - Major	USA	265.83B	21.05	94.32	0.31%	8,874,201
8	GE	General Electric Company	Industrial Goods	Diversified Machinery	USA	262.24B	17.44	25.99	-1.14%	48,660,488
9	WFC	Wells Fargo & Company	Financial	Money Center Banks	USA	245.96B	12.00	46.67	0.37%	11,812,453
10	WMT	Wal-Mart Stores Inc.	Services	Discount, Variety Stores	USA	245.41B	14.52	75.35	-0.65%	5,853,718
11	RDS-B	Royal Dutch Shell plc	Basic Materials	Major Integrated Oil & Gas	Netherlands	237.98B	11.24	75.79	1.36%	3,268,235
12	CVX	Chevron Corporation	Basic Materials	Major Integrated Oil & Gas	USA	230.51B	9.86	120.43	0.06%	4,974,472
13	HSBC	HSBC Holdings plc	Financial	Foreign Money Center Banks	United Kingdom	223.00B	13.00	55.75	0.04%	1,157,459
14	PG	Procter & Gamble Co.	Consumer Goods	Personal Products	USA	216.71B	20.11	79.23	-1.18%	8,455,602
15	JPM	JPMorgan Chase & Co.	Financial	Money Center Banks	USA	216.66B	13.21	57.59	-1.00%	15,432,152
16	PFE	Pfizer Inc.	Healthcare	Drug Manufacturers - Major	USA	205.79B	21.57	31.27	0.13%	22,347,414
17	CHL	China Mobile Limited	Technology	Wireless Communications	Hong Kong	202.57B	9.53	50.39	0.52%	655,292
18	IBM	International Business Machines Corp.	Technology	Information Technology Services	USA	198.82B	12.60	182.25	-3.28%	13,785,619
19	NVS	Novartis AG	Healthcare	Drug Manufacturers - Major	Switzerland	197.29B	21.78	80.79	-0.71%	1,244,071
20	TM	Toyota Motor Corporation	Consumer Goods	Auto Manufacturers - Major	Japan	192.81B	14.22	121.69	1.02%	371,366

야후 파이낸스 (종목 검색기)

http://screener.finance.yahoo.com/stocks.html



기타 (종목 검색기)

- **Google Finance**
  http://www.google.com/finance/stockscreener
- **Morningstar**
  http://screen.morningstar.com/stockselector.html
- **MarketWatch**
  http://www.marketwatch.com/tools/stockresearch/screener/

미국 시장 주요 종목

미국 시장(AMEX, NYSE, NASDAQ) 시가총액 상위 20 (2014년 1월, 출처: http://finviz.com/)

No.	Ticker	Company	Sector	Industry	Country
1	AAPL	Apple Inc.	Consumer Goods	Electronic Equipment	USA
2	XOM	Exxon Mobil Corporation	Basic Materials	Major Integrated Oil & Gas	USA
3	GOOG	Google Inc.	Technology	Internet Information Providers	USA
4	MSFT	Microsoft Corporation	Technology	Business Software & Services	USA
5	PTR	PetroChina Co. Ltd.	Basic Materials	Major Integrated Oil & Gas	China
6	BRK-A	Berkshire Hathaway Inc.	Financial	Property & Casualty Insurance	USA
7	JNJ	Johnson & Johnson	Healthcare	Drug Manufacturers - Major	USA
8	GE	General Electric Company	Industrial Goods	Diversified Machinery	USA
9	WFC	Wells Fargo & Company	Financial	Money Center Banks	USA
10	WMT	Wal-Mart Stores Inc.	Services	Discount, Variety Stores	USA

한국 거래소 종목 심볼

국내지수

- ^KS11 : KOSPI composite Index (거래소 지수)
- ^KQ11 : KOSDAQ composite Index (코스닥 지수)

국내 종목

- 종목코드에 .KS 가 붙는다. 예를 들어, 삼성전자의 심볼은 005930.KS

시가총액기준 상위 20위 종목 (2014.1 기준)

순위	종목 심볼	종목명		순위	종목 심볼	종목명
1	005930.KS	삼성전자		11	032830.KS	삼성생명
2	005380.KS	현대차		12	051910.KS	LG화학
3	005490.KS	POSCO		13	009540.KS	현대중공업
4	012330.KS	현대모비스		14	017670.KS	SK텔레콤
5	000660.KS	SK하이닉스		15	105560.KS	KB금융
6	035420.KS	NAVER		16	096770.KS	SK이노베이션
7	005935.KS	삼성전자우		17	023530.KS	롯데쇼핑
8	015760.KS	한국전력		18	086790.KS	하나금융지주
9	055550.KS	신한지주		19	000810.KS	삼성화재
10	000270.KS	기아차		20	066570.KS	LG전자

참고가 될만한 심볼들

외환

- XAUUSD=X : 국제 금 고시가격 (달러/온스)
- USDKRW=X : 원달러 환율

미국 뉴욕증권시장(NYSE)의 3대 지수: 다우지수, 나스닥, S&P500

- ^DJI : Down Jones Industrial Average (다우존스 지수)
- ^IXIC : NASDAQ composite (나스닥 지수)
- ^GSPC : S&P 500 Index (S&P 500 지수)

국가별 주요 지수

- ^N225 : Nikkei 225 (일본 니케이 지수)
- ^HSI : Hang Seng Index (홍콩 항셍 지수)
- ^HSCE : Hang Seng China Enterprises Index (홍콩 중국기업 지수 (H주))
- 000001.SS : SSE composite (상해 지수)
- ^BSESN : Bombay Sensex Index (인도 지수)
- ^BVSP : IBOVESPA (브라질 지수)
- ^MXX : IPC (멕시코 지수)
- RTS.RS : RTSI Index (러시아 지수)

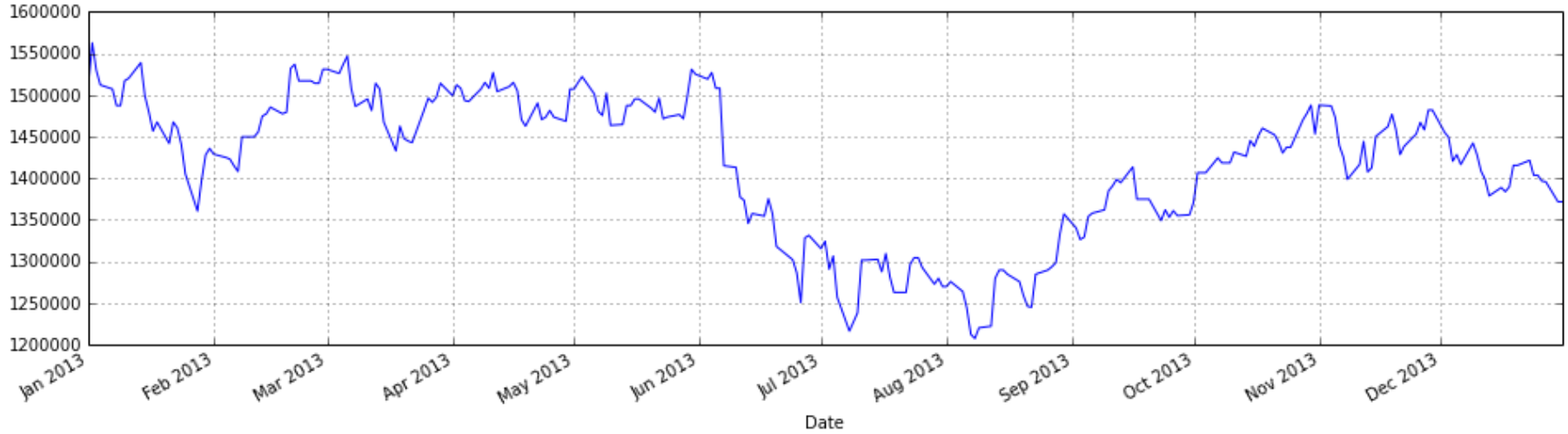
종가Close 와 수정종가Adj Close

수정주가(**Adjusted Closing Price**): 분할, 배당, 배분, 신주 발행이 반영된 주식 가격

데이터의 연속성을 보장하기 때문에 데이터 분석에는 **"Adj Close"** 사용한다.

삼성전자(005930.KS) 2013년 일별 주가

```
import requests
import pandas as pd
from StringIO import StringIO

url = 'http://real-chart.finance.yahoo.com/table.csv?' \
      's=005930.KS&a=0&b=1&c=2013&d=11&e=31&f=2013&g=d'

r = requests.get(url)
df = pd.read_csv(StringIO(r.content), index_col='Date', parse_dates={'Date'})
df['Adj Close'].plot(figsize=(16, 4))
```


1600000
1550000
1500000
1450000
1400000
1350000
1300000
1250000
1200000
Jan 2013
Feb 2013
Mar 2013
Apr 2013
May 2013
Jun 2013
Jul 2013
Aug 2013
Sep 2013
Oct 2013
Nov 2013
Dec 2013
Date

요청 URL 만들기

http://real-chart.finance.yahoo.com/table.csv?s=005930.KS&a=0&b=1&c=2013&d=11&e=31&f=2013&g=d

키	설명	값	비고
s	종목(심볼)	005930.KS	Samsung Electronics Co. Ltd.
a	시작 월	0	1월 (0부터 시작)
b	시작 일	1	
c	시작 년	2013	
d	끝 월	11	12월 (0부터 시작)
e	끝 일	31	
f	끝 년	2013	
g	기간	d:일, w :주, m:월	v:'배당'만 표시

```
url = 'http://real-chart.finance.yahoo.com/table.csv?' \
      's=005930.KS&a=0&b=1&c=2013&d=11&e=31&f=2013&g=d'

r = requests.get(url)
lines = r.content.splitlines()
lines[:10]
```

```
['Date,Open,High,Low,Close,Volume,Adj Close',
 '2013-12-31,1372000.00,1372000.00,1372000.00,1372000.00,000,1371554.72',
 '2013-12-30,1396000.00,1397000.00,1360000.00,1372000.00,338100,1371554.72',
 '2013-12-27,1410000.00,1411000.00,1395000.00,1396000.00,210200,1395546.93',
 '2013-12-26,1408000.00,1415000.00,1401000.00,1408000.00,246100,1396506.61',
 '2013-12-25,1415000.00,1415000.00,1415000.00,1415000.00,000,1403449.47',
 '2013-12-24,1430000.00,1432000.00,1415000.00,1415000.00,219600,1403449.47',
 '2013-12-23,1437000.00,1438000.00,1429000.00,1433000.00,212700,1421302.54',
 '2013-12-20,1413000.00,1427000.00,1413000.00,1427000.00,168100,1415351.52',
 '2013-12-19,1418000.00,1434000.00,1418000.00,1427000.00,211000,1415351.52']
```

```
df = pd.read_csv(StringIO(r.content))
df.head()
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	2013-12-31	1372000	1372000	1372000	1372000	0	1371554.72
1	2013-12-30	1396000	1397000	1360000	1372000	338100	1371554.72
2	2013-12-27	1410000	1411000	1395000	1396000	210200	1395546.93
3	2013-12-26	1408000	1415000	1401000	1408000	246100	1396506.61
4	2013-12-25	1415000	1415000	1415000	1415000	0	1403449.47

```
pd.to_datetime(df['Date']) #  datetime 컬럼으로 변환
df = df.set_index('Date') # 인덱스 컬럼 지정
df.head()
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2013-12-31	1372000	1372000	1372000	1372000	0	1371554.72
2013-12-30	1396000	1397000	1360000	1372000	338100	1371554.72
2013-12-27	1410000	1411000	1395000	1396000	210200	1395546.93
2013-12-26	1408000	1415000	1401000	1408000	246100	1396506.61
2013-12-25	1415000	1415000	1415000	1415000	0	1403449.47

DataFrame.read_csv() 에서 인덱스와 시간 문자열 지정

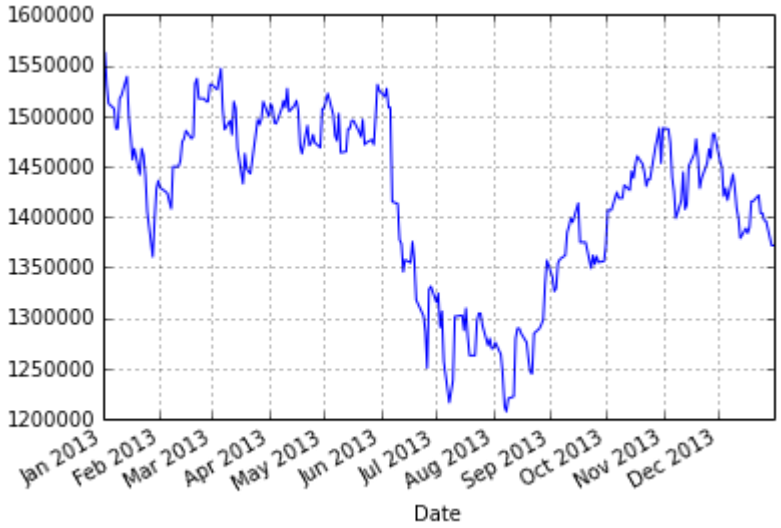
```
DataFrame.read_csv(f, index_col='Date', parse_dates={'Date'})
```

```
url = 'http://real-chart.finance.yahoo.com/table.csv?' \
      's=005930.KS&a=0&b=1&c=2013&d=11&e=31&f=2013&g=d'

r = requests.get(url)
df = pd.read_csv(StringIO(r.content), index_col='Date', parse_dates={'Date'})
df.head()
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2013-12-31	1372000	1372000	1372000	1372000	0	1371554.72
2013-12-30	1396000	1397000	1360000	1372000	338100	1371554.72
2013-12-27	1410000	1411000	1395000	1396000	210200	1395546.93
2013-12-26	1408000	1415000	1401000	1408000	246100	1396506.61
2013-12-25	1415000	1415000	1415000	1415000	0	1403449.47

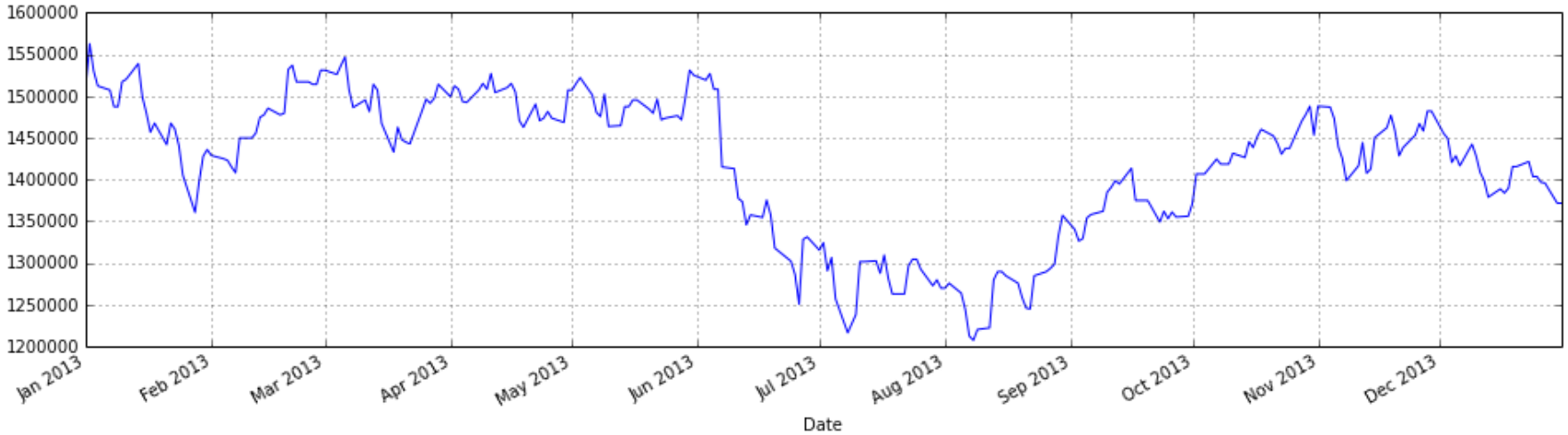
DataFrame.plot()

```
df['Adj Close'].plot()
```



DataFrame.plot() 크기 지정

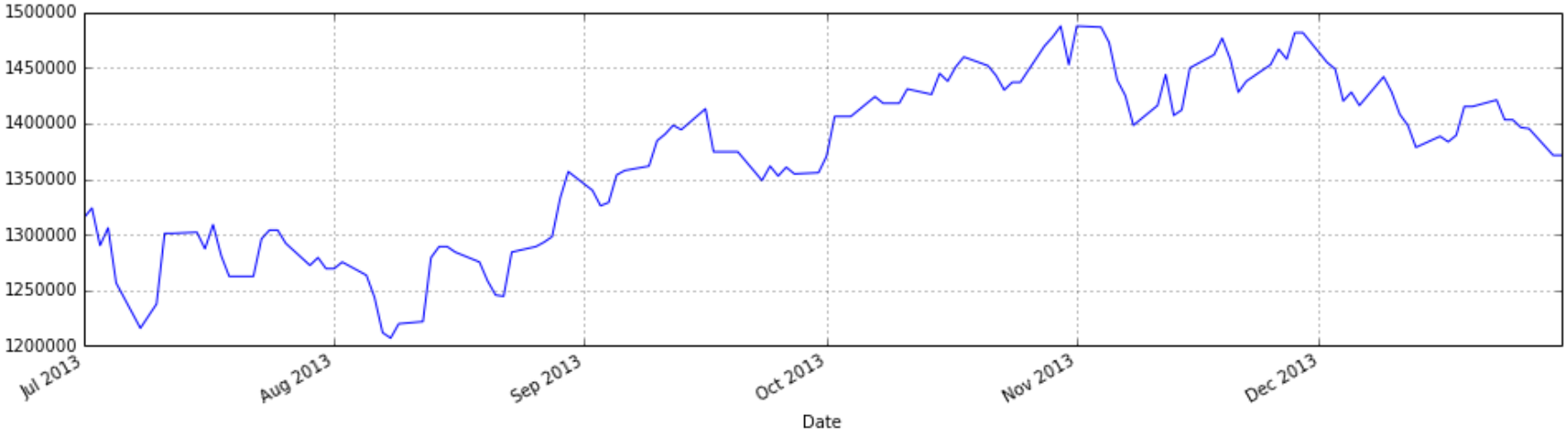
```
df['Adj Close'].plot(figsize=(16, 4))
```



기간을 지정하는 다양한 방법들

- df['2013'] # 2013년
- df['2013-06'] # 6월
- df['2013-07-01':'2013-09-30'] # 3/4 분기
- df[:'2013-06-30'] # 상반기

```
df['2013-07-01':]['Adj Close'].plot(figsize=(16, 4))  # 하반기
```

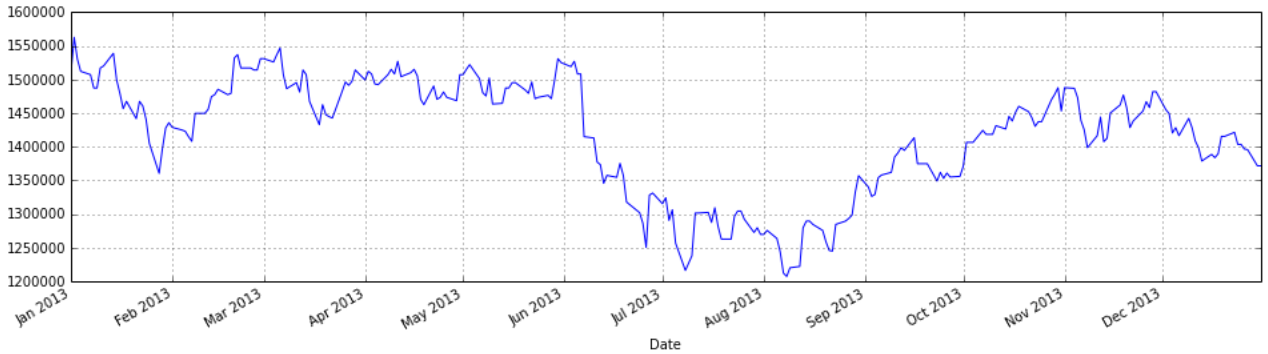


모아서 살펴보기

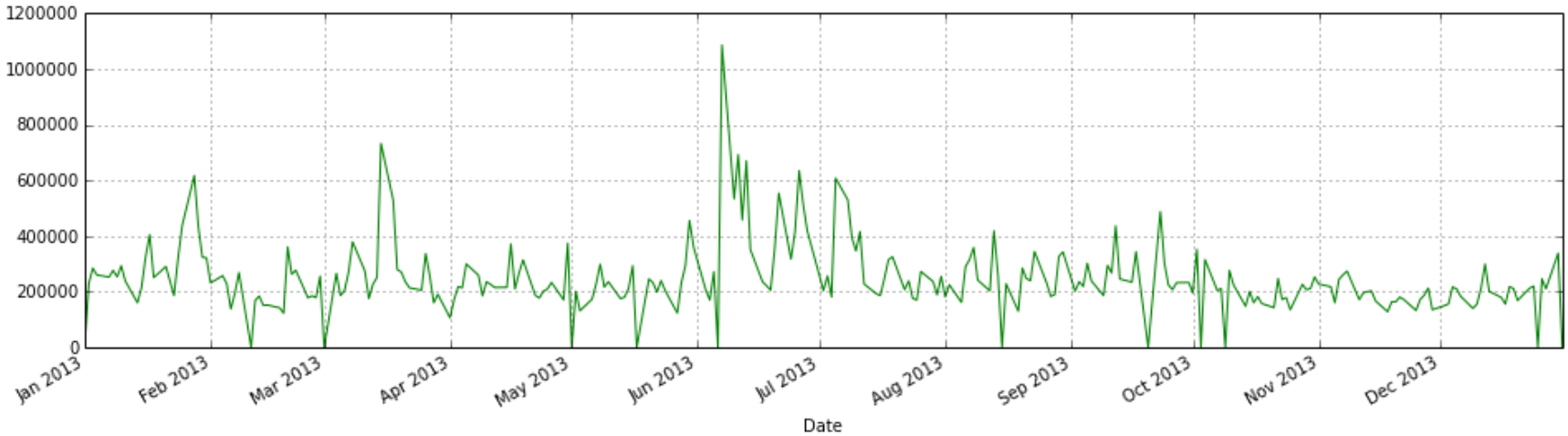
야후 파이낸스 CSV (pandas, requests, StringIO 사용)

1. requests.get(url)
2. pandas.read_csv(file)
3. DataFrame.plot()

```
import requests
import pandas as pd
from StringIO import StringIO

url = 'http://real-chart.finance.yahoo.com/table.csv?' \
      's=005930.KS&a=0&b=1&c=2013&d=11&e=31&f=2013&g=d'

r = requests.get(url)
df = pd.read_csv(StringIO(r.content), index_col='Date', parse_dates={'Date'})
df['Adj Close'].plot(figsize=(16, 4))
```



```
df['Volume'].plot(figsize=(16, 4), style='g')
```



5. 수익률과 분포

수익률: 투자한 자본에 대한 수익(혹은 손실)의 비율

$$\text{수익률} = \frac{(\text{나중가격} - \text{처음가격})}{\text{처음가격}}$$

- 예를 들어, 100원을 투자해서 110원이 됐다면 수익률은 +10%
- 금융 분석에서는 가격을 쓰지 않고 수익률을 쓴다

한번 더 생각해 보기

하루는 10% 이익을, 하루는 10% 손해를 봤다면 수익률이 0% 일까?

1000원 ⟶ +10% (잔액 1100원) ⟶ -10% (잔액 990원)

수익과 손실의 순서를 바꾸면?

1000원 ⟶ -10% (잔액 900원) ⟶ +10% (잔액 990원)

일반수익률과 로그수익률

$$\text{일반 수익률}(R) = \frac{\text{나중가격} - \text{처음가격}}{\text{처음가격}} = \frac{P - P_0}{P_0} = \frac{P}{P_0} - 1$$

$$\text{로그 수익률}(R) = \ln \frac{\text{나중가격}}{\text{처음가격}} = \ln \frac{P}{P_0} = \ln(P) - \ln(P_0)$$

일반수익률과 로그수익률 비교

거래일	가격	일반수익률	로그수익률
1일	1,000	.	.
2일	1,300	30.00%	26.24%
3일	800	-38.46%	-48.55%
4일	1,300	62.50%	48.55%
5일	1,100	-15.38%	-16.71%
수익률 합계	39%	38.65%	9.53%
최종수익률	10%	10.00%	9.53%

일반 수익률

- 재투자 해서 발생하는 손실까지 포함
- 각 거래에서 발생하는 손실률의 합과 최종 손실률이 달라진다

로그 수익률

- 최종 수익률과 수익률의 합계가 일치
- 금융 분야에서는 주로 로그 수익률을 사용

로그 수익률 계산, 2013년 삼성전자(005930.KS) 일일 주가

```
import pandas as pd
from datetime import datetime
from pandas.io.data import DataReader

start = datetime(2013, 1, 1)
end = datetime(2013, 12, 30)

df = DataReader("005930.KS", "yahoo", start, end)

# 거래량('Volume')이 0인 row 제거
df = df[df['Volume'] != 0]
```

```
df['Adj Close'].head()
```

```
Date
2013-01-02    1562602.25
2013-01-03    1529882.78
2013-01-04    1512035.80
2013-01-07    1507078.31
2013-01-08    1487248.33
Name: Adj Close, dtype: float64
```

```
# shift(1), 데이터를 1씩 이동
df['Adj Close'].shift(1).head()
```

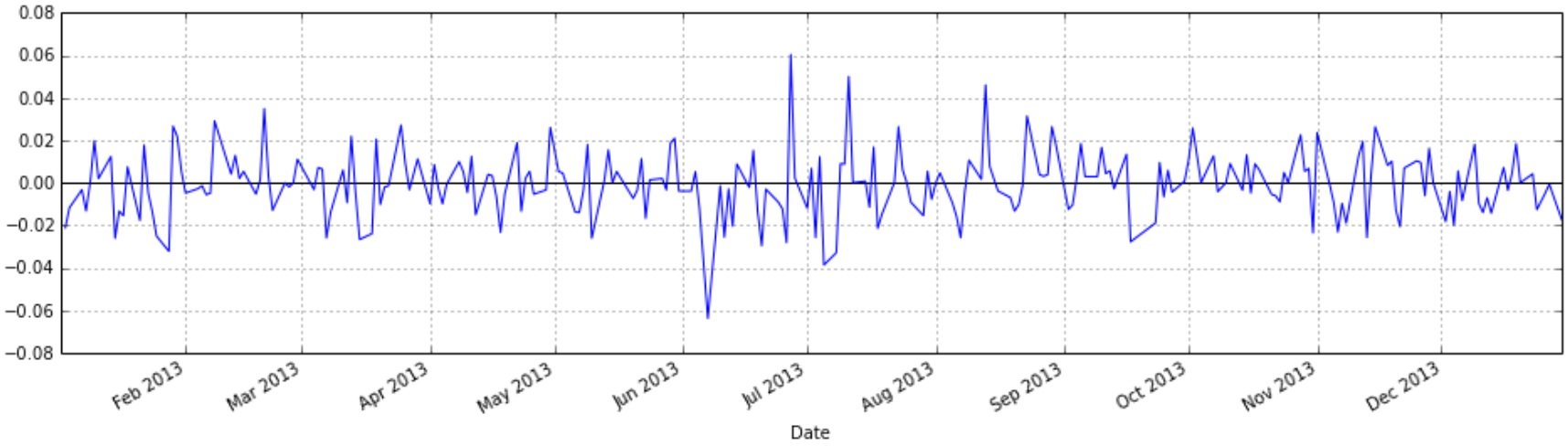
```
Date
2013-01-02          NaN
2013-01-03    1562602.25
2013-01-04    1529882.78
2013-01-07    1512035.80
2013-01-08    1507078.31
Name: Adj Close, dtype: float64
```

```
df['Ret'] = log( df['Adj Close'] / df ['Adj Close'].shift(1) )
df.head()
```

	Open	High	Low	Close	Volume	Adj Close	Ret
Date							
2013-01-02	1533000	1576000	1527000	1576000	228900	1562602.25	NaN
2013-01-03	1582000	1584000	1543000	1543000	284500	1529882.78	-0.021161
2013-01-04	1540000	1542000	1510000	1525000	259900	1512035.80	-0.011734
2013-01-07	1515000	1528000	1500000	1520000	252200	1507078.31	-0.003284
2013-01-08	1513000	1517000	1498000	1500000	276400	1487248.33	-0.013245

수익률의 변동

```
plt.axhline(color='k')
df['Ret'].plot(figsize=(16, 4))
```



기본 통계량 - 대푯값 representative value

- 평균(mean): mean() - 주가, 기대수익률, 기댓값
- 중앙값(median): median() - 많이 사용X
- 최빈값(mode): mode() - 많이 사용X

기본 통계량 - 산포도 measure of dispersion

- 왜도(Skewness), 첨도(kurtosis): skew(), kurt() - 확률분포의 특성, 시장의 심리
- 분산(variance): var() - 리스크, 변동성
- 표준편차(standard deviation): std() - 투자위험(risk), 매매신호

DataFrame 기본 통계량 함수

```
# 대푯값
df.mean() # 산술평균
df.median() # 중앙값
df.mode() # 최빈값

# 산포도
df.skew() # 왜도:  평균으로 부터 왼쪽/오른쪽으로 치우친 정도  (0 정상, - 오른쪽, +왼쪽)
df.kurt() # 첨도: 표준(정규분포) 위로 표족, 납작한 정도 (> 3 뾰족, <3 납작, =3 표준)
df.var() # 분산: 평균에서 떨어진 정도(제곱)
df.std() # 표준편차: 평균에서 떨어진 정도
```

DataFrame.describe()

- 자주 사용하는 통계 요약값들
- 개수, 평균, 표준편차, 최소, 최대, 사분위 값

```
df.describe()
```

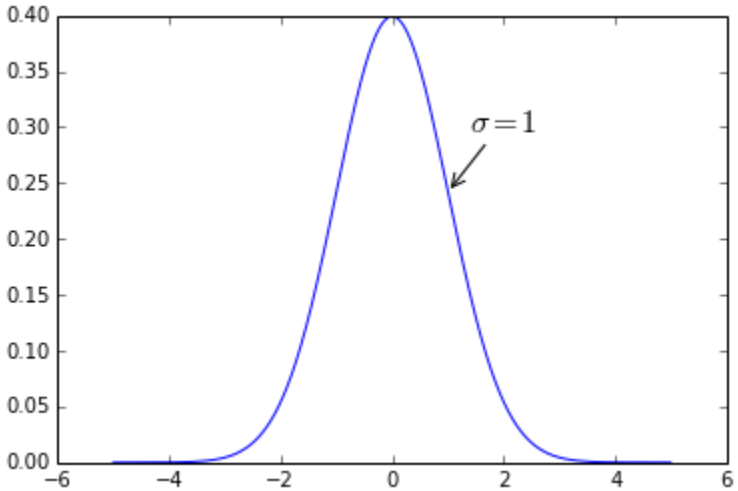
	Open	High	Low	Close	Volume	Adj Close	Ret
count	247.00	247.00	247.00	247.00	247.00	247.00	246.0000
mean	1430866.39	1442380.56	1416012.14	1428927.12	257614.17	1417105.18	-0.0005
std	87991.65	86379.48	86147.70	86616.35	116934.19	85711.37	0.0152
min	1213000.00	1235000.00	1209000.00	1217000.00	105900.00	1207065.73	-0.0637
25%	1369500.00	1378500.00	1359000.00	1369000.00	186350.00	1357361.98	-0.0096
50%	1451000.00	1461000.00	1434000.00	1450000.00	228300.00	1438163.77	0.0000
75%	1500000.00	1509000.00	1484000.00	1497500.00	284550.00	1484769.58	0.0073
max	1582000.00	1584000.00	1548000.00	1576000.00	1085800.00	1562602.25	0.0603

정규확률분포함수

Normal Probability Distribution Function, Normal PDF

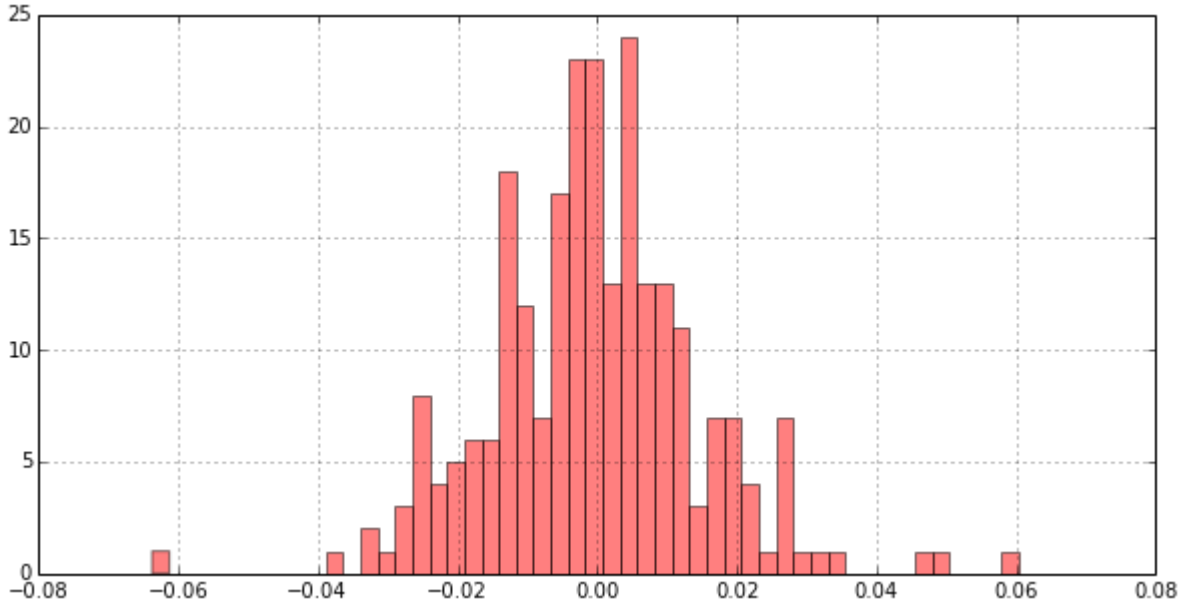
y = mlab.normpdf(bins, mu, sigma)

```
r = np.arange(-5, 5, 0.01)
mu = 0
sigma = 1
plt.plot(r, mlab.normpdf(r, mu, sigma), color='b')

plt.annotate(r'$\sigma=1$', xy=(1, mlab.normpdf(1, mu, sigma)),
        xytext=(+10, +30), textcoords='offset points', fontsize=16,
        arrowprops=dict(arrowstyle="->"))
```



수익률의 분포

```
df['Ret'].hist(bins=50, color='r', alpha=0.5, figsize=(10,5))
```



수익률 분포와 정규분포

- 수익률의 확률분포는 정규분포와 유사 (통계적 특성 분석 용이)
- 정규분포와 차이 - 첨도가 크고(더 뾰족하고), 두꺼운 꼬리(fat tail)
- 정규분포와 비교해 크게 오르거나, 떨어지는 현상이 더 나타남
  (예, 외환 위기, 금융 위기)
- 정규분포와 유사하기 때문에 실무적으로는 정규분포를 가정

6. 이동평균과 시그널

pandas 이동평균

- pandas.stats.rolling_mean()

```
df['MA_5'] = pd.stats.moments.rolling_mean(df['Adj Close'], 5)
df['MA_20'] = pd.stats.moments.rolling_mean(df['Adj Close'], 20)
df['diff'] = df['MA_5'] - df['MA_20']
df.head(10)
```

	Open	High	Low	Close	Volume	Adj Close	MA_5	MA_20	diff
Date									
2013-01-01	1522000	1522000	1522000	1522000	0	1509061.31	NaN	NaN	NaN
2013-01-02	1533000	1576000	1527000	1576000	228900	1562602.25	NaN	NaN	NaN
2013-01-03	1582000	1584000	1543000	1543000	284500	1529882.78	NaN	NaN	NaN
2013-01-04	1540000	1542000	1510000	1525000	259900	1512035.80	NaN	NaN	NaN
2013-01-07	1515000	1528000	1500000	1520000	252200	1507078.31	1524132.090	NaN	NaN
2013-01-08	1513000	1517000	1498000	1500000	276400	1487248.33	1519769.494	NaN	NaN
2013-01-09	1500000	1513000	1491000	1500000	253100	1487248.33	1504698.710	NaN	NaN
2013-01-10	1515000	1534000	1500000	1530000	293200	1516993.30	1502120.814	NaN	NaN
2013-01-11	1548000	1548000	1507000	1533000	238200	1519967.79	1503707.212	NaN	NaN
2013-01-14	1539000	1552000	1528000	1552000	159900	1538806.27	1510052.804	NaN	NaN

골든크로스, 데드크로스

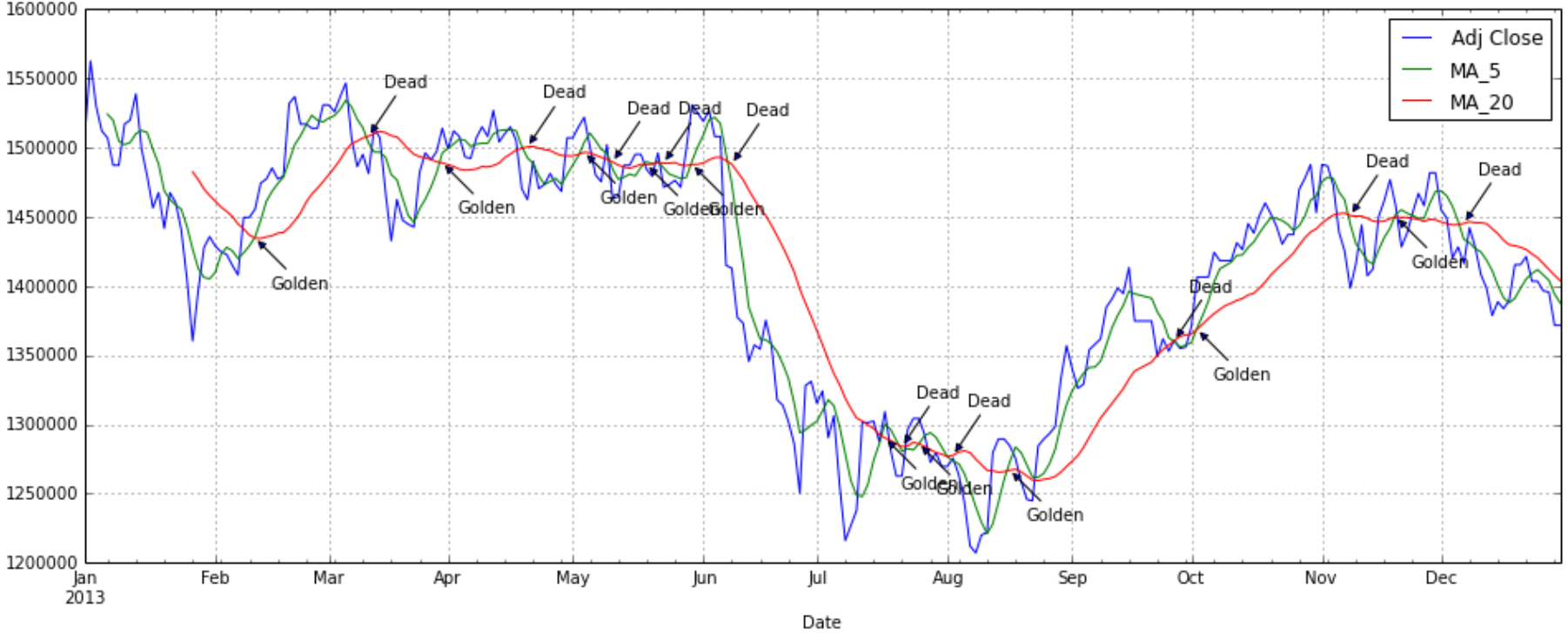
단기와 장기 이동평균의 차이값(MA_5 - MA_20)를 비교

크로스: 차이값 x 이전 차이값 < 0 (즉, 이전 값과 부호가 바뀌는 경우)

```
prev_key = prev_val = 0

for key, val in df['diff'][1:].iteritems():
    if val == 0:
        continue
    if val * prev_val < 0 and val > prev_val:
        print '[golden]', key, val
    if val * prev_val < 0 and val < prev_val:
        print '[dead]', key, val
    prev_key, prev_val = key, val
```

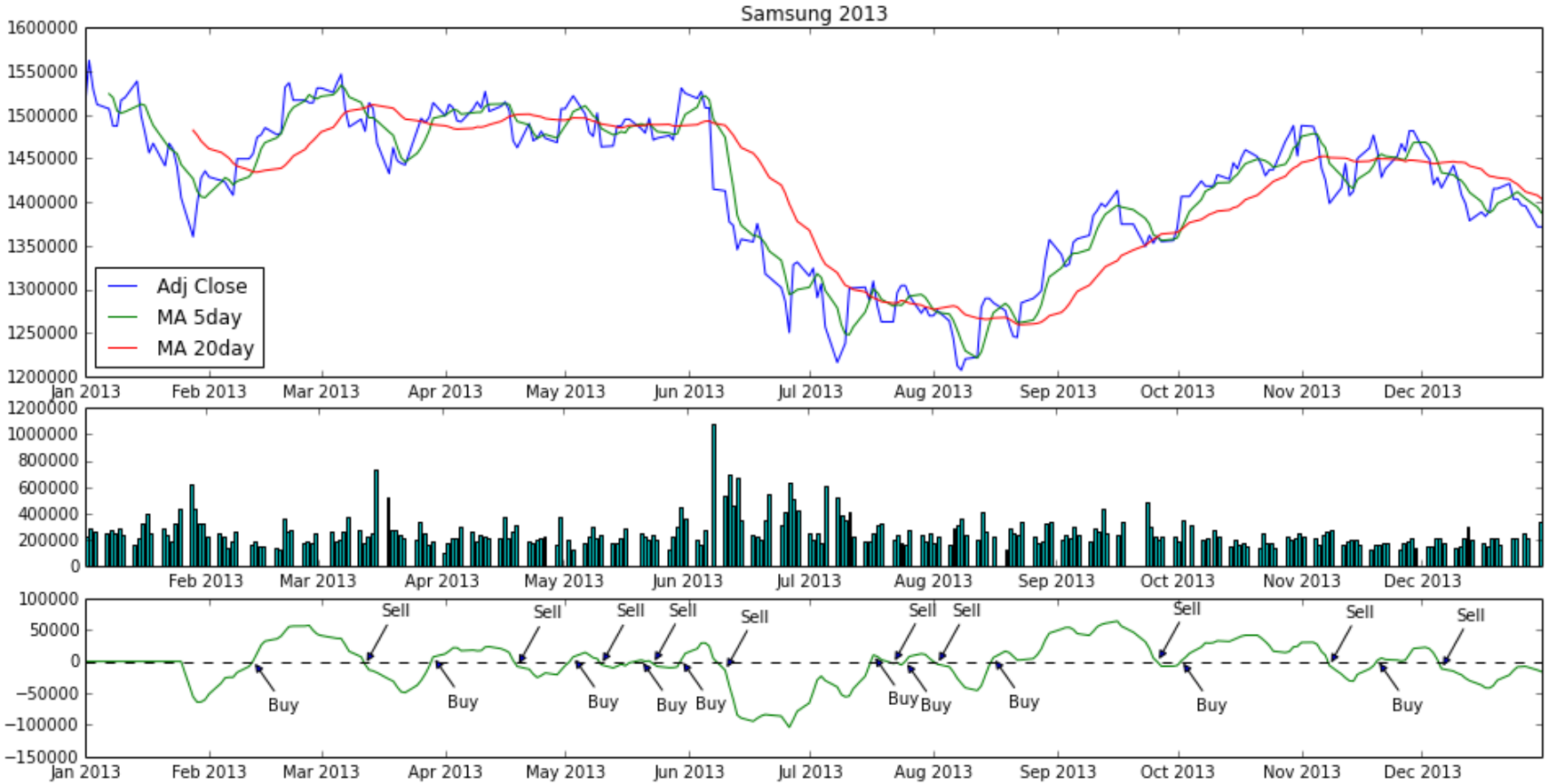
```
[golden] 2013-02-12 00:00:00 842.774000001
[dead] 2013-03-12 00:00:00 -4957.4955
[golden] 2013-03-29 00:00:00 7237.9425
[dead] 2013-04-19 00:00:00 -7882.416
[golden] 2013-05-03 00:00:00 7287.5155
... ...
```

```
ax = df[['Adj Close', 'MA_5', 'MA_20']].plot(figsize(16,6))
prev_key = prev_val = 0

for key, val in df['diff'][1:].iteritems():
    if val == 0:
        continue

    if val * prev_val < 0 and val > prev_val:
        ax.annotate('Golden', xy=(key, df['MA_20'][key]), xytext=(10,-30),
                    textcoords='offset points', arrowprops=dict(arrowstyle='-|>'))
    if val * prev_val < 0 and val < prev_val:
        ax.annotate('Dead', xy=(key, df['MA_20'][key]), xytext=(10,30),
                    textcoords='offset points', arrowprops=dict(arrowstyle='-|>'))

    prev_key, prev_val = key, val
```


Adj Close
MA_5
MA_20
Dead
Golden
1600000
1550000
1500000
1450000
1400000
1350000
1300000
1250000
1200000
Jan
2013
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Date

```
fig = matplotlib.pyplot.gcf()
fig.set_size_inches(16, 8)

# price (가격)
price_chart = plt.subplot2grid((4,1), (0, 0), rowspan=2)
price_chart.plot(df.index, df['Adj Close'], label='Adj Close')
price_chart.plot(df.index, df['MA_5'], label='MA 5day')
price_chart.plot(df.index, df['MA_20'], label='MA 20day')

plt.title(u'Samsung 2013')
plt.legend(loc='best')

# volume (거래량)
vol_chart = plt.subplot2grid((4,1), (2,0), rowspan=1)
vol_chart.bar(df.index, df['Volume'], color='c')
```

```
# 이동평균의 차이
signal_chart = plt.subplot2grid((4,1), (3,0), rowspan=1)
signal_chart.plot(df.index, df['diff'].fillna(0), color='g')
plt.axhline(y=0, linestyle='--', color='k')

prev_key = prev_val = 0 # sell, buy annotate

for key, val in df['diff'][1:].iteritems():
    if val == 0:
        continue
    if val * prev_val < 0 and val > prev_val:
        signal_chart.annotate('Buy', xy=(key, df['diff'][key]), xytext=(10,-30),
textcoords='offset points', arrowprops=dict(arrowstyle='-|>'))
    if val * prev_val < 0 and val < prev_val:
        signal_chart.annotate('Sell', xy=(key, df['diff'][key]), xytext=(10,30),
textcoords='offset points', arrowprops=dict(arrowstyle='-|>'))
    prev_key, prev_val = key, val
```


Samsung 2013
Adj Close
MA 5day
MA 20day
1600000
1550000
1500000
1450000
1400000
1350000
1300000
1250000
1200000
Jan 2013
Feb 2013
Mar 2013
Apr 2013
May 2013
Jun 2013
Jul 2013
Aug 2013
Sep 2013
Oct 2013
Nov 2013
Dec 2013
1200000
1000000
800000
600000
400000
200000
0
100000
50000
0
−50000
−100000
−150000
Buy
Sell

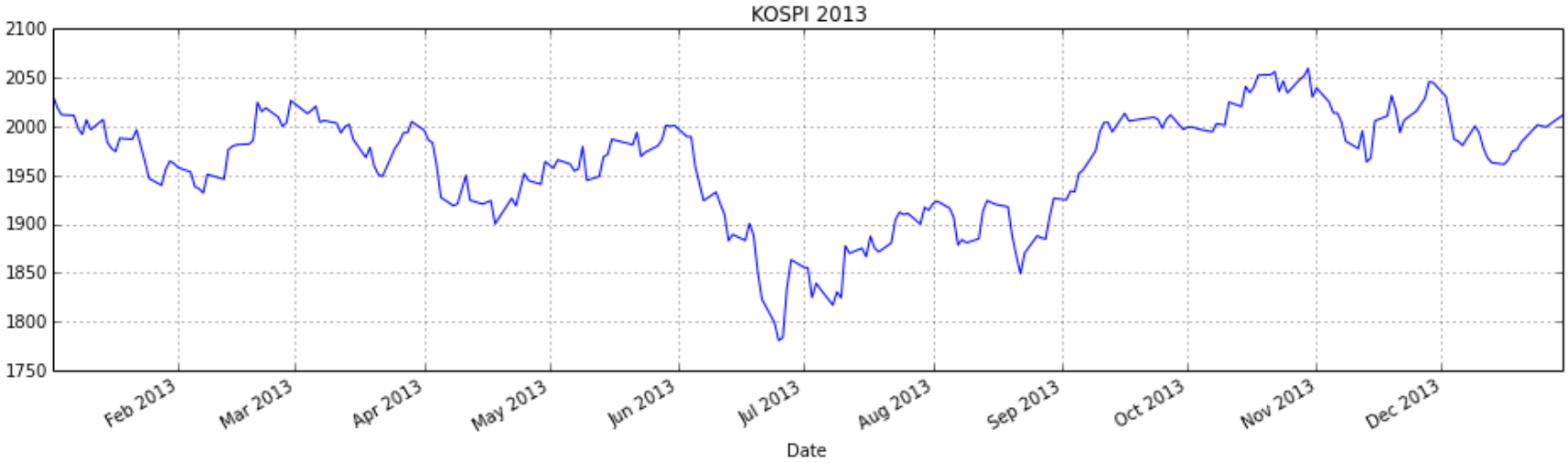
7. 지수와 거래량

데이터 준비

```
import requests
import pandas as pd
from StringIO import StringIO

url = 'http://real-chart.finance.yahoo.com/table.csv?' \
      's=^KS11&a=0&b=1&c=2013&d=11&e=31&f=2013&g=d'

r = requests.get(url)
df = pd.read_csv(StringIO(r.content), index_col='Date', parse_dates={'Date'})
```

가격 Adj Close

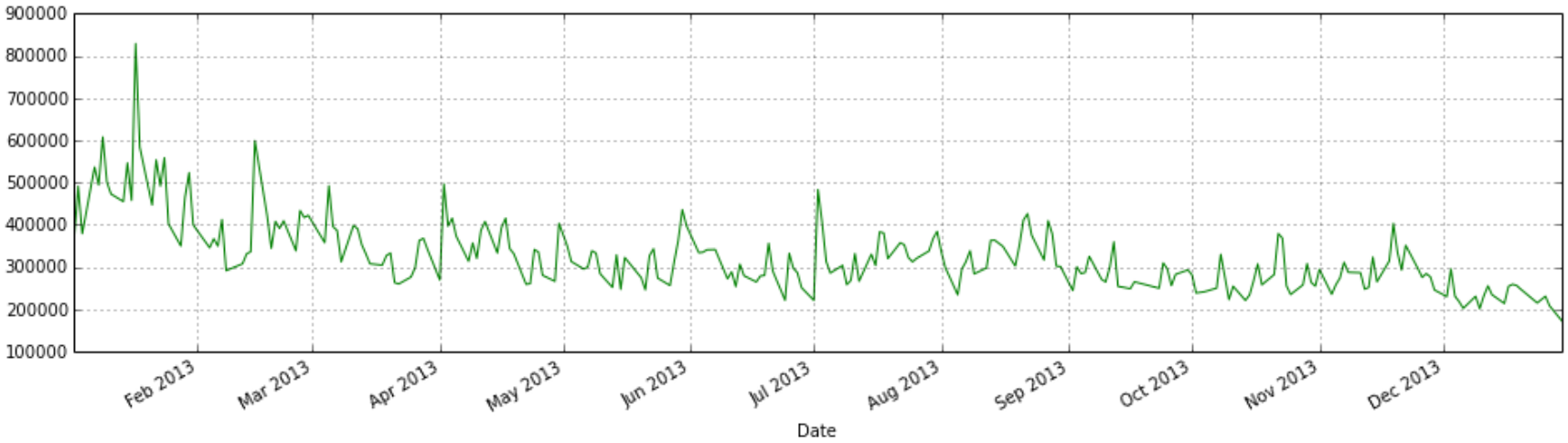
```
df['Adj Close'].plot(figsize=(16, 4), title='KOSPI 2013')
```



거래량 Volume

1. 일반적으로 거래량은 주가변화에 선행
2. 일반적으로 거래량이 늘면 주가는 상승하고 거래량이 줄면 주가는 하락
3. 거래량에 급격한 변화가 있으면 주가가 급등하거나 급락

```
df['Volume'].plot(figsize=(16, 4), style='g')
```

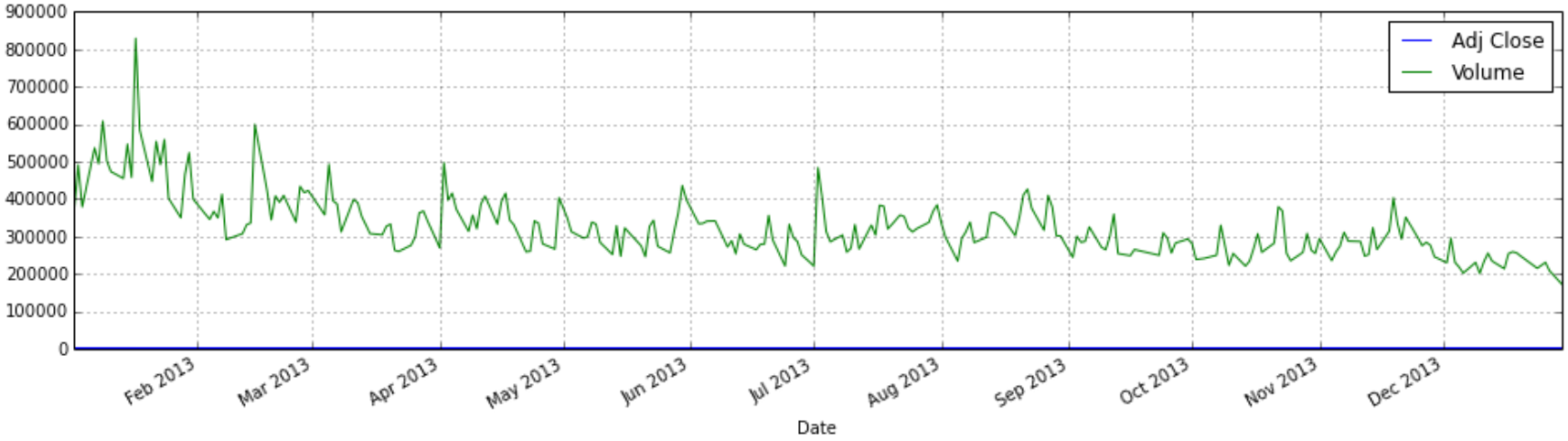


거래량과 주가비교

```
df2 =  df[['Adj Close', 'Volume']]
df2.head()
```

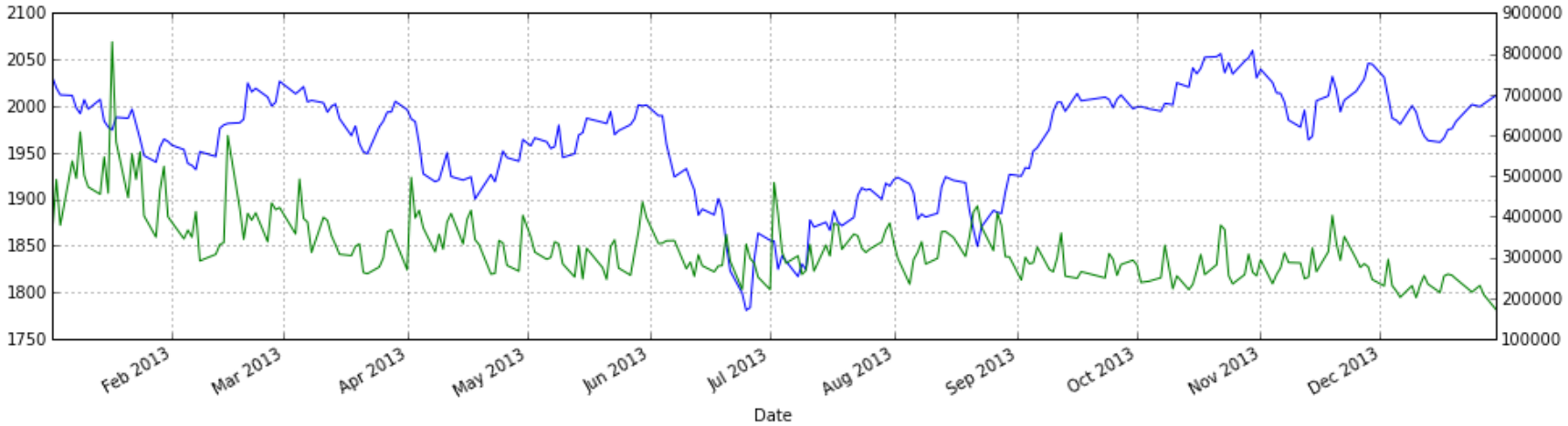
	Adj Close	Volume
Date		
2013-12-30	2011.34	172200
2013-12-27	2002.28	207800
2013-12-26	1999.30	230500
2013-12-24	2001.59	214700
2013-12-23	1996.89	223000

```
df2.plot(figsize=(16, 4))
```

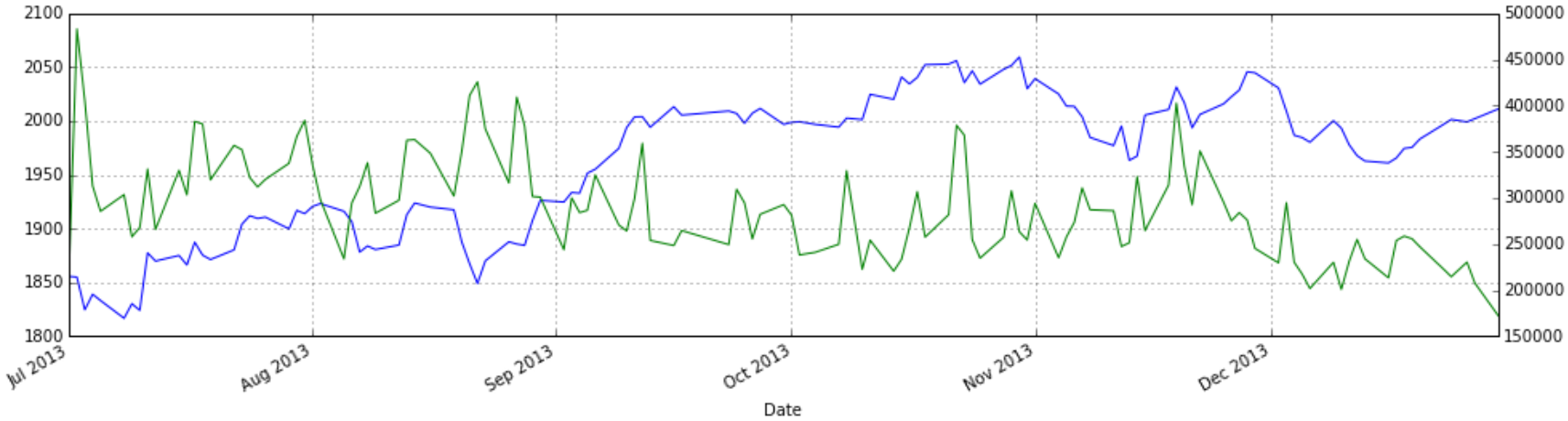


두 번째 Y축 Secondary Y

```
df2['Adj Close'].plot(figsize=(16, 4), style='b')
df2['Volume'].plot(figsize=(16, 4), style='g', secondary_y=True)
```



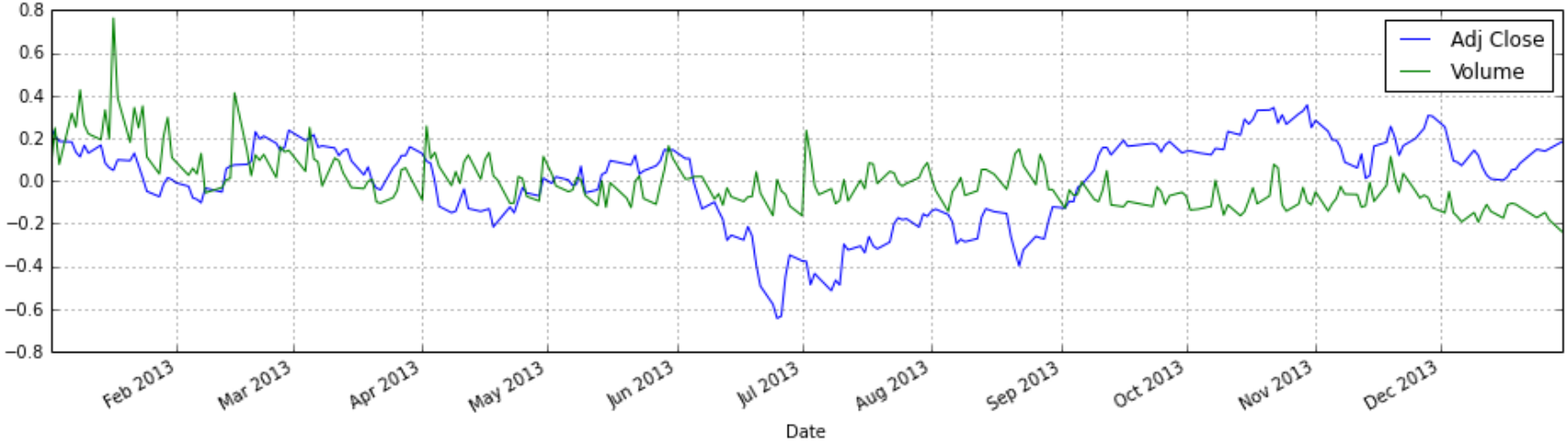
```
df2['2013-07':]['Adj Close'].plot(figsize=(16, 4), style='b')
df2['2013-07':]['Volume'].plot(figsize=(16, 4), style='g', secondary_y=True)
```



정규화 Normalize

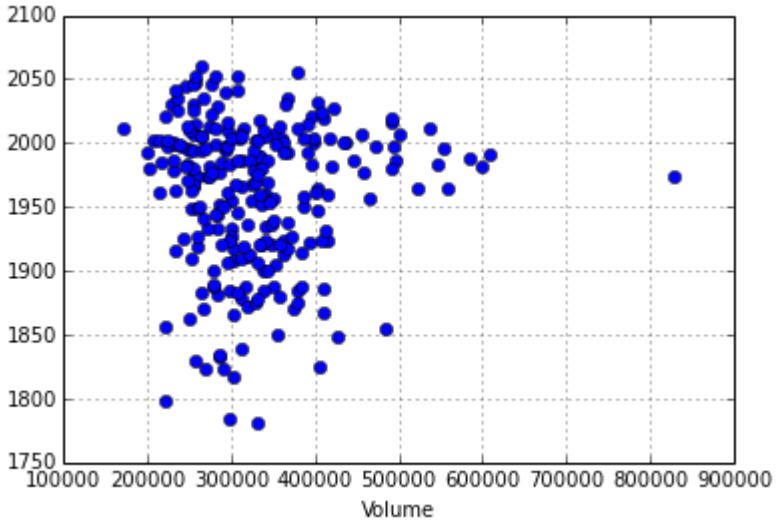
- 시계열 데이터를 0~1 사이의 값으로 변환
- 데이터의 변화를 상대적으로 비교

```
df_norm = (df2 - df2.mean()) / (df2.max() - df2.min())
df_norm.plot(figsize=(16, 4))
```



스캐터 차트

```
df2.plot(y='Adj Close', x='Volume', style='o')
```



상관계수

```
df2.corr()
```

	Adj Close	Volume
Adj Close	1.000000	0.007143
Volume	0.007143	1.000000

8. 상관분석

상관분석 Correlation Analysis

두 변수간에 어떤 선형적 관계를 갖고 있는 지를 분석하는 방법

- 공분산: (음수, 0, 양수)
- 상관계수: (0 ~ 1 사이의 값)
- 베타: (몇 배인지 표현)

(데이터 집합 A와 B에 대해)

공분산: (음수, 0, 양수)

- 양수면 A가 커짐에 따라 B도 커진다는 것을 의미
- (음수, 0, 양수인지만 중요, 값의 크기는 중요하지 않음)

상관계수: (0 ~ 1 사이의 값)

- 양수면 양의 상관관계, 음수면 음의 상관관계

베타: (몇 배인지 표현)

- A값이 움직일 때, B값이 얼마나(몇 배) 움직이는지를 표현
- 시장베타: 종목의 가격이 시장 KOSPI 200 의 변화에 얼마나 영향을 받는지

2013년 12월 기준 시가총액 상위 10개사

순위	코드	회사이름	시가총액	시가총액 비중	산업
1	005930	삼성전자	202.095	15.47%	반도체와반도체장비
2	005380	현대차	52.095	3.99%	자동차
3	012330	현대모비스	28.570	2.19%	자동차부품
4	005490	POSCO	28.467	2.18%	철강
5	000660	SK하이닉스	26.135	2.00%	반도체와반도체장비
6	035420	NAVER	23.865	1.83%	인터넷소프트웨어와서비스
7	005935	삼성전자우	23.130	1.77%	반도체와반도체장비
8	000270	기아차	22.741	1.74%	자동차
9	055550	신한지주	22.430	1.72%	은행
10	015760	한국전력	22.308	1.71%	전기유틸리티

여러 종목의 가격

```
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
from pandas.io.data import DataReader

top10_codes = [
    '005930.KS','005380.KS','012330.KS','005490.KS','000660.KS',
    '035420.KS','005935.KS','000270.KS','055550.KS','015760.KS' ]

top5_codes = [
    '005930.KS','005380.KS','012330.KS','005490.KS','000660.KS' ]
```

```
start = datetime(2013, 1, 1)
end  = datetime(2013, 12, 31)

df = DataReader(top5_codes, 'yahoo', start=start, end=end)
df = df['Adj Close']

df.head()
```

	000660.KS	005380.KS	005490.KS	005930.KS	012330.KS
Date					
2013-01-01	25750	216988.43	339371.62	1509061.31	286325.91
2013-01-02	26600	214505.72	350554.35	1562602.25	285828.81
2013-01-03	26650	204574.90	359792.26	1529882.78	270418.91
2013-01-04	26350	204574.90	356875.03	1512035.80	259482.85
2013-01-07	25900	207057.61	358819.85	1507078.31	261968.32

컬럼 순서 바꾸기

```
df = df[top5_codes]
df.head()
```

	005930.KS	005380.KS	012330.KS	005490.KS	000660.KS
Date					
2013-01-01	1509061.31	216988.43	286325.91	339371.62	25750
2013-01-02	1562602.25	214505.72	285828.81	350554.35	26600
2013-01-03	1529882.78	204574.90	270418.91	359792.26	26650
2013-01-04	1512035.80	204574.90	259482.85	356875.03	26350
2013-01-07	1507078.31	207057.61	261968.32	358819.85	25900

컬럼 이름 바꾸기

- Dataframe.rename()

```
code_names = {
    '005930.KS':'Samsung Elec', '005380.KS':'Hyundai Motor',
    '012330.KS':'Hyundai Mobis', '005490.KS':'POSCO',
    '000660.KS':'SK Hynix','035420.KS':'Naver',
    '005935.KS':'Samsung Elec(Prep)', '000270.KS':'Kia Motor',
    '055550.KS':'Shinhan', '015760.KS':'Korea Elc Pwr' }

df = df.rename(columns=code_names)
df.head()
```

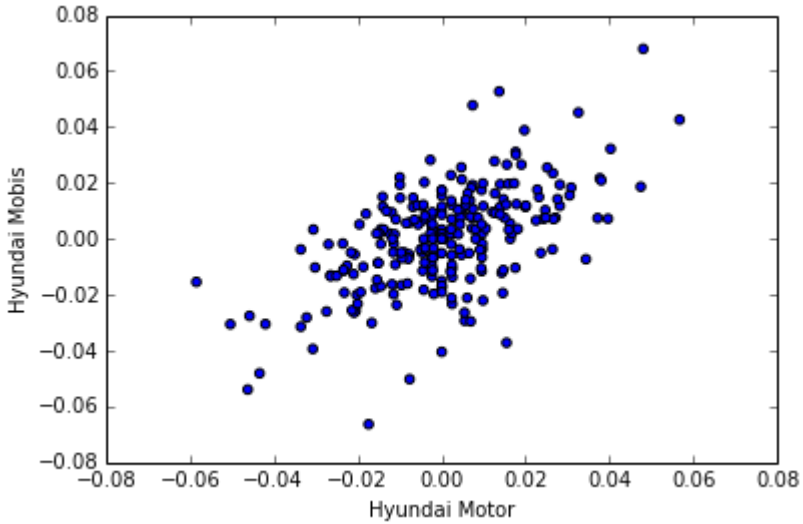
	Samsung Elec	Hyundai Motor	Hyundai Mobis	POSCO	SK Hynix
Date					
2013-01-01	1509061.31	216988.43	286325.91	339371.62	25750
2013-01-02	1562602.25	214505.72	285828.81	350554.35	26600
2013-01-03	1529882.78	204574.90	270418.91	359792.26	26650
2013-01-04	1512035.80	204574.90	259482.85	356875.03	26350
2013-01-07	1507078.31	207057.61	261968.32	358819.85	25900

등락률 (전일대비)

```
changes = df.pct_change()
changes.head()
```

	Samsung Elec	Hyundai Motor	Hyundai Mobis	POSCO	SK Hynix
Date					
2013-01-01	NaN	NaN	NaN	NaN	NaN
2013-01-02	0.035480	-0.011442	-0.001736	0.032951	0.033010
2013-01-03	-0.020939	-0.046296	-0.053913	0.026352	0.001880
2013-01-04	-0.011666	0.000000	-0.040441	-0.008108	-0.011257
2013-01-07	-0.003279	0.012136	0.009579	0.005450	-0.017078

```
plt.scatter(changes['Hyundai Motor'], changes['Hyundai Mobis'])
plt.xlabel('Hyundai Motor')
plt.ylabel('Hyundai Mobis')
```



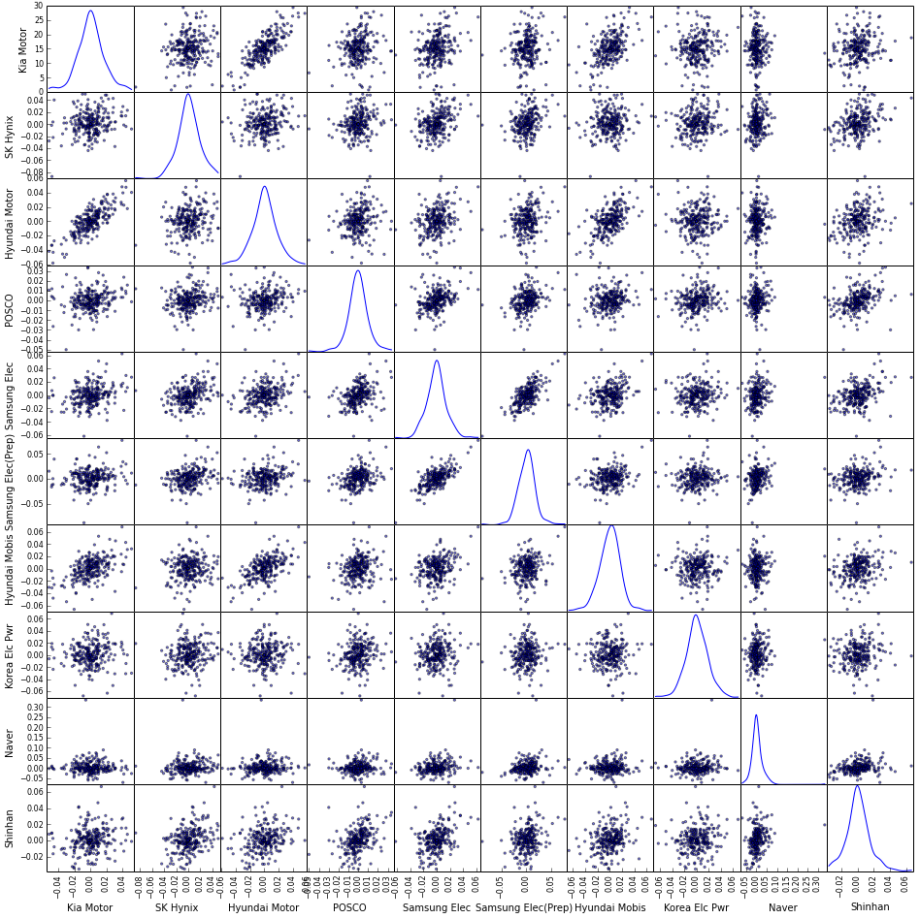
상관계수

```
corr = changes.corr()
corr
```

	Samsung Elec	Hyundai Motor	Hyundai Mobis	POSCO	SK Hynix
Samsung Elec	1.000000	0.320175	0.321344	0.357618	0.342223
Hyundai Motor	0.320175	1.000000	0.576877	0.167033	0.192207
Hyundai Mobis	0.321344	0.576877	1.000000	0.190388	0.122958
POSCO	0.357618	0.167033	0.190388	1.000000	0.187668
SK Hynix	0.342223	0.192207	0.122958	0.187668	1.000000

상위 10개 종목 상관관계 차트

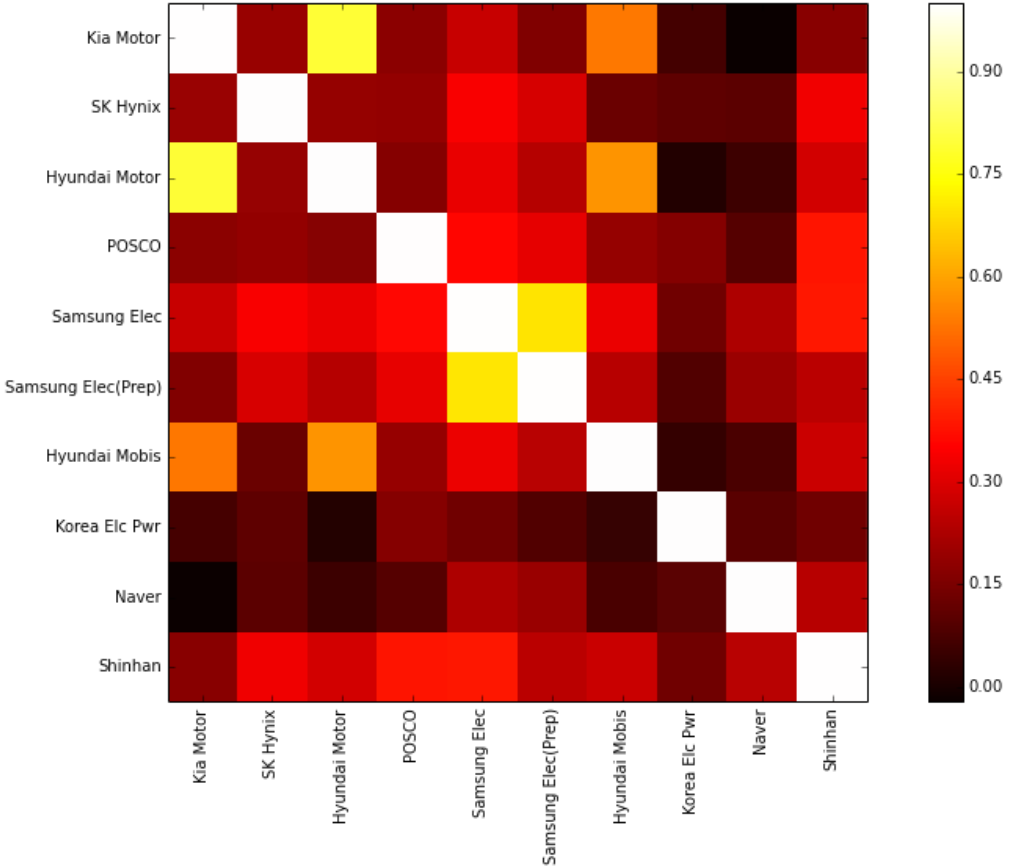
```
df = DataReader(top10_codes, 'yahoo', start=start, end=end)
df = df['Adj Close']
df = df.rename(columns=code_names)

changes = df.pct_change()
pd.scatter_matrix(changes, diagonal='kde', figsize=(16, 16));
```



상위 10개 종목 상관관계 스캐터 차트

상위 10개 종목 상관관계 히트맵

```
corr = changes.corr()

plt.figure(figsize=(14,8))
plt.imshow(corr, cmap='hot', interpolation='none')
plt.colorbar()
plt.xticks(range(len(corr)), corr.columns, rotation=90)
plt.yticks(range(len(corr)), corr.columns)
plt.show()
```



상위 10개 종목 상관관계 히트맵

상관계수 순위

```
idx = []; vals = []
for ix, i in enumerate(corr.columns.values):
    for j in corr.columns.values[ix + 1:]:
        idx.append((i, j))
        vals.append(corr[i][j])

ser = pd.Series(data=vals, index=idx)
ser_ord = ser.order(ascending=False)
ser_ord[:10]
```

(Kia Motor, Hyundai Motor)	0.793340
(Samsung Elec, Samsung Elec(Prep))	0.702924
(Hyundai Motor, Hyundai Mobis)	0.576877
(Kia Motor, Hyundai Mobis)	0.533587
(Samsung Elec, Shinhan)	0.388561
(POSCO, Shinhan)	0.384025
(POSCO, Samsung Elec)	0.357618
(SK Hynix, Samsung Elec)	0.342223
(SK Hynix, Shinhan)	0.329163
(Samsung Elec, Hyundai Mobis)	0.321344

- 현대차, 기아차 0.792788
- (현대자동차, 기아자동차, 현대모비스)는 서로 강한 상관관계
- 강한 음의 상관관계는 나타나지 않았다

종목간 상관계수

```
code_names = { '^KS11':'KOSPI',
               '005930.KS':'Samsung Elec', '005380.KS':'Hyundai Motor',
               '012330.KS':'Hyundai Mobis', '005490.KS':'POSCO',
               '000660.KS':'SK Hynix','035420.KS':'Naver',
               '005935.KS':'Samsung Elec(Prep)', '000270.KS':'Kia Motor',
               '055550.KS':'Shinhan', '015760.KS':'Korea Elc Pwr' }

df = DataReader(code_names.keys(), 'yahoo', start='2013-01-01', end='2013-12-31')
df = df['Adj Close']
df = df.rename(columns=code_names)

changes = df.pct_change()
chg_corr = changes.corr()
chg_corr
```

	Kia Motor	SK Hynix	Hyundai Motor	POSCO	Samsung Elec	Samsung Elec(Prep)	Hyundai Mobis	Korea Elc Pwr	Naver	Shinhan	KOSPI
Kia Motor	1.000	0.194	0.793	0.177	0.267	0.158	0.534	0.068	-0.022	0.172	0.446
SK Hynix	0.194	1.000	0.192	0.188	0.342	0.292	0.123	0.109	0.102	0.329	0.459
Hyundai Motor	0.793	0.192	1.000	0.167	0.320	0.239	0.577	0.017	0.057	0.284	0.525
POSCO	0.177	0.188	0.167	1.000	0.358	0.314	0.190	0.169	0.093	0.384	0.544
Samsung Elec	0.267	0.342	0.320	0.358	1.000	0.703	0.321	0.135	0.225	0.389	0.770
Samsung Elec(Prep)	0.158	0.292	0.239	0.314	0.703	1.000	0.243	0.089	0.200	0.248	0.569
Hyundai Mobis	0.534	0.123	0.577	0.190	0.321	0.243	1.000	0.043	0.074	0.271	0.471
Korea Elc Pwr	0.068	0.109	0.017	0.169	0.135	0.089	0.043	1.000	0.101	0.135	0.233
Naver	-0.022	0.102	0.057	0.093	0.225	0.200	0.074	0.101	1.000	0.242	0.305
Shinhan	0.172	0.329	0.284	0.384	0.389	0.248	0.271	0.135	0.242	1.000	0.659
KOSPI	0.446	0.459	0.525	0.544	0.770	0.569	0.471	0.233	0.305	0.659	1.000

KOSPI 와 다른 종목간 상관계수

```
ser = chg_corr['KOSPI']
ser_ord = ser.order(ascending=False)
ser_ord[1:]
```

```
Samsung Elec         0.770216
Shinhan              0.659464
Samsung Elec(Prep)   0.569390
POSCO                0.543671
Hyundai Motor        0.524798
Hyundai Mobis        0.470791
SK Hynix             0.459246
Kia Motor            0.446092
Naver                0.304662
Korea Elc Pwr        0.232799
Name: KOSPI, dtype: float64
```

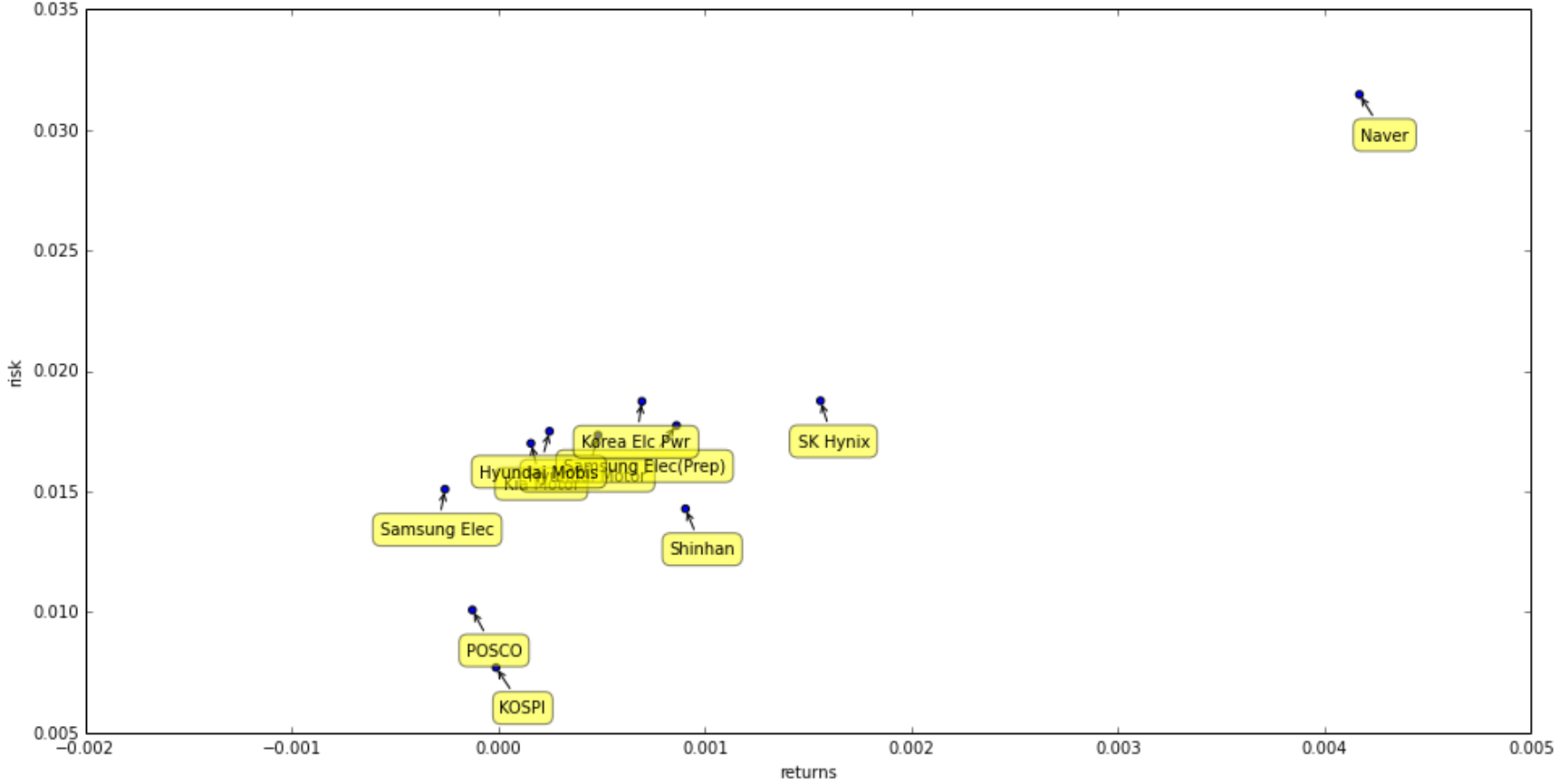
- 상관계수 순위 Series.order()
- 삼성전자, 신한, 삼성전자(우), POSCO, 현대차 순

수익과 위험

- 수익 returns: 수익률 평균 mean
- 위험 risk: 표준편차 std, 값이 클수록 변동성이 크므로 위험이 크다

수익성returns와 위험 Risk 차트

```
plt.figure(figsize=(16,8))
plt.scatter(changes.mean(), changes.std())
plt.xlabel('returns')
plt.ylabel('risk')

for label, x, y in zip(changes.columns, changes.mean(), changes.std()):
    plt.annotate( label,xy=(x, y), xytext=(30, -30),
        textcoords = 'offset points', ha = 'right', va = 'bottom',
        bbox = dict(boxstyle = 'round,pad=0.5', fc = 'yellow', alpha = 0.5),
        arrowprops = dict(arrowstyle = '->', connectionstyle = 'arc3,rad=0'))
```


0.035
0.030
0.025
0.020
0.015
0.010
0.005
risk
−0.002
−0.001
0.000
0.001
0.002
0.003
0.004
0.005
returns
Naver
SK Hynix
Korea Elc Pwr
Hyundai Mobis
Samsung Elec(Prep)
Samsung Elec
Shinhan
POSCO
KOSPI

9. 뱀발 (蛇足)

2014 세계수학자대회(ICM)

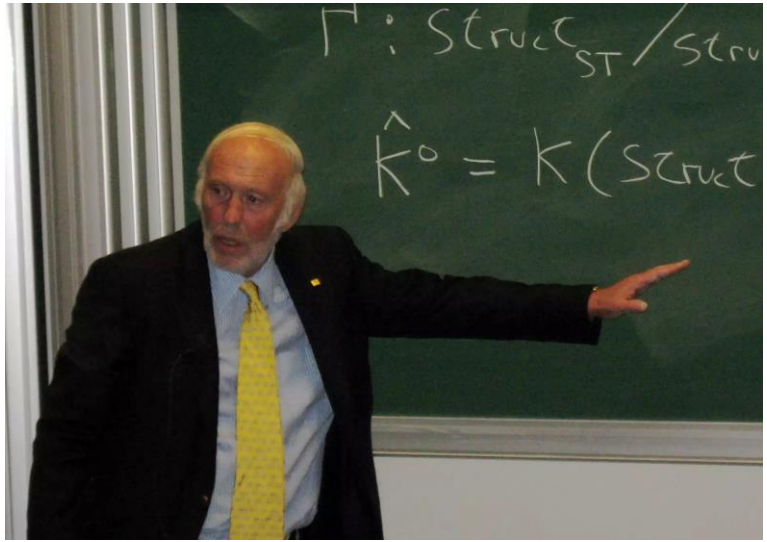
8월 13일 - 21일

삼성동 코엑스

초청연사 제임스 사이먼스

제임스 사이먼스

- 미국의 수학자, 억만장자, 퀀트 펀드 창시자
- MIT를 거쳐 UC버클리에서 미분기하학으로 박사학위
- 1970년 월가에 진출, 수학자, 통계학자 등과 르네상스 테크놀로지 설립(1982)
- 세계 억만장자 88위 (자산 약 13조원)



제임스 사이먼스의 투자법

"미래를 예측하지 말고 현재에 집중 하라"

- 많은 주식 투자자들은 미래를 예상해 투자. 사이먼스는 이는 "틀렸다"고 단언
- "가장 기본적인 원리는 '시장은 시시각각 변한다'는 것입니다"

개인도 데이터 수집/분석/응용이 가능한 시대 !

예측이 아니라, 대응!

절대적인 확신을 찾지 말고 가능성 위주로 탐구하는
확률적 사고가 중요.

이승준 (PyCon 2014 Korea)
fb.com/plusjune